Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models*

Johannes Brumm
DBF, University of Zurich
johannes.brumm@uzh.ch

Simon Scheidegger
DBF, University of Zurich
simon.scheidegger@uzh.ch

January 15, 2014

Abstract

We present a flexible and scalable method to compute global solutions of high-dimensional stochastic dynamic models. Within a time-iteration setup, we interpolate policy functions using an adaptive sparse grid algorithm with piecewise multi-linear (hierarchical) basis functions. As the dimensionality increases, sparse grids grow considerably slower than standard tensor product grids. In addition, the grid scheme we use is automatically refined locally and can thus capture steep gradients or even non-differentiabilities. To further increase the maximum problem size we can handle, our implementation is fully hybrid parallel, i.e. using a combination of distributed and shared memory parallelization schemes. This parallelization enables us to efficiently use high-performance computing architectures. Our algorithm scales up nicely to more than one thousand parallel processes. To demonstrate the performance of our method, we apply it to high-dimensional international real business cycle models with capital adjustment costs and irreversible investment.

Keywords: Adaptive Sparse Grids, High-Performance Computing, International Real Business Cycles, Occasionally Binding Constraints

JEL Classification: C63, C68, F41

*We are very grateful to Felix Kübler for helpful discussions and support. We thank Ken Judd, Karl Schmedders and seminar participants at University of Zürich, Stanford University, University of Chicago, Argonne National Laboratory, and CEF 2013 in Vancouver for valuable comments. Moreover, we thank Xiang Ma for very instructive email discussions regarding the workings of adaptive sparse grids. We are grateful for the support of Olaf Schenk, Antonio Messina and Riccardo Murri concerning HPC related issues. We acknowledge CPU time granted on the University of Zürich's 'Schrödinger' HPC cluster. Johannes Brumm gratefully acknowledges financial support from the ERC.

1 Introduction

There are many important economic phenomena that cannot be captured by models if they do not account for the interactions between different firms, sectors, and countries, or if they only consider local dynamics around steady states. At the latest, this has become obvious through the recent financial crisis with its tremendous spillover effects and large price fluctuations. Yet already in the nineties, more and more economists included various kinds of heterogeneity in their models and also started to use global solution techniques (see, e.g. [36] or [23]). However, solving for the global solution of a model with substantial heterogeneity is very costly: Using conventional solution methods, the computation time and storage requirements increase exponentially with the amount of heterogeneity, i.e. with the dimensionality of the problem.

This paper makes an effort to shift the limits of how much heterogeneity we can assume in an economic model and still be able to compute an accurate global solution in a reasonable amount of time. We achieve this by employing a highly parallel implementation of a so-called adaptive sparse grid method within a time iteration framework. This method can handle high-dimensional problems even if they exhibit non-smooth behavior like non-differentiable policy functions.

Standard algorithms that are used to compute global solutions of economic models rely on a grid-based numerical representation of a multi-dimensional policy function (see, [18]). However, starting with a one-dimensional discretization scheme that employs N gridpoints, a straightforward extension to d dimensions leads to N^d gridpoints. Sparse grids are able to alleviate this so-called 'curse of dimensionality' by reducing the number of gridpoints from the order $\mathcal{O}\left(N^d\right)$ to $\mathcal{O}\left(N \cdot (\log N)^{d-1}\right)$ with only slightly deteriorated accuracy if the underlying function is sufficiently smooth (see, e.g. [5], with references therein).

The sparse grid construction we are using was introduced by Zenger [39] for the solution of partial differential equations. However, the underlying principle, a sparse tensor product decomposition, goes back to the seminal work of Smolyak [35]. Sparse grids have been applied to a whole range of different research fields such as physics, visualization, finance and econometrics (see, e.g. [10, 5, 14, 26, 38]). Using the original formulation of Smolyak [35], Krüger and Kübler [22] were the first to solve dynamic economic models using sparse grids. Recently, Judd et al. [17] proposee an implementation that is more efficient and also allows for grids that are ex ante chosen to be finer in some dimensions than in others. However, these two papers rely on global polynomials as basis functions, which fail to capture the local behavior of policy functions that are not sufficiently smooth. In contrast, our algorithm can capture non-smooth behaviour. The reason is that we use hierarchical basis functions with an adaptive grid refinement strategy [24]. The basis functions we use are hat functions, which are piecewise multi-linear functions with local support. The space of basis functions is hierarchically structured into interpolation levels. For a basis function at a given level l, there are several basis functions of the next finer level $l+1$ among which the support of the original function of level l is subdivided. Using the value of a level l function, an automatic grid adaptation strategy decides whether the interpolation is refined locally by adding the associated $l+1$ functions. Importantly, this refinement scheme scales just linearly with increasing dimension (see, e.g. [24, 29, 28]). Such an adaptive sparse grid with hierarchical local basis functions offers the promise of an efficient and accurate solution of

economic problems that are both high-dimensional and non-smooth.

However, the latter class of problems requires substantial computation time even if an efficient solution method is applied. Therefore, our implementation aims to access high-performance computing (HPC) facilities. Their mainstream hardware design nowadays consists of shared memory nodes with several multi-core CPUs that are connected via a network structure. Hence, efficient parallel programming must make use of multiple computational units by combining distributed memory parallelization on the node interconnect with shared memory parallelization inside each node (see, e.g. [30]). We address this challenge by an implementation that is 'hybrid' parallel, i.e. using MPI ('Message Passing Interface'; cf., [34]) between nodes and OpenMP (shared memory parallelism; cf., [16]) within the nodes. In the hybrid MPI/OpenMP mode, we are able to efficiently use at least 1,200 cores.

To demonstrate that our algorithm can solve standard high-dimensional economic problems, we solve the international real business cycle (IRBC) model with adjustment costs. For this application the performance of alternative algorithms is well documented in a study comparing various solution methods (see, [21]). Like many of these established methods, we use a time iteration procedure (see, e.g. [18]) to solve for an equilibrium that is recursive in the capital stock and the productivity levels of all countries. The innovation of our approach lies within each time iteration step, where we use an (adaptive) sparse grid to interpolate the policy functions. As the policy functions in this model are very smooth, our linear interpolation scheme has a disadvantage compared to smooth interpolation schemes. Nevertheless, we can compute quite accurate solutions for models that are of higher dimension than any that have been reported in the comparison study by [21]. However, the purpose and comparative advantage of our algorithm lies in solving models that exhibit non-smooth behavior. To demonstrate its performance with respect to such models, we augment the IRBC model with irreversible investment. In spite of the non-differentiabilities (also called 'kinks') induced by this assumption we are still able to compute accurate solutions for high-dimensional examples. The adaptivity of the grid now ensures that we can capture the kinks fairly well without increasing the number of gridpoints too much.

The main contribution of this paper is as follows: We apply for the first time an adaptive sparse grid algorithm to solve large-scale economic models. Using modern high-performance computing facilities, we are able to compute accurate global solutions for models with up to 24 dimensions, and also to high-dimensional models with kinks.

In addition to the above discussed literature on sparse grids, both in mathematics and economics, our paper is also closely related to two other strands of the literature. First, our work is related to papers that develope methods for solving dynamic economic models with occasionally binding constraints (see, e.g. [15, 3, 9, 1]). While these methods are able to match the non-differentiabilities induced by such constraints very precisely for up to three continuous state variables, they are not as flexible and scalable as ours. The adaptive sparse grid technique we propose is therefore superior when it comes to problems with more than three dimensions. It can at least handle problems with occasionally binding constraints that have up to eight-dimensional state spaces. Second, our paper is part of the emergent literature on parallel computing applications in economics (see, e.g. [6]). To the best of our knowledge, we are the first to ef-

ficiently use current high-performance computing technology to solve dynamic economic models. We are able to do so as our implementation is fully hybrid parallel.

The remainder of the paper is organized as follows. In Sec. 2, we explain the construction of adaptive sparse grids and also provide simple test cases for their use in interpolation. In Sec. 3, we embed adaptive sparse grid interpolation in a time iteration algorithm to solve high-dimensional IRBC models, also with irreversible investment. We then discuss the performance of this algorithm and report how hybrid parallelization can speed up the computations. Sec. 4 concludes.

2 From Full Grids to Adaptive Sparse Grids

In this section, we first provide a brief introduction to 'classical', i.e. non-adaptive, sparse grid interpolation. In contrast to the sparse grid interpolation schemes employed so far in economics (see, e.g. [22, 17]), which rely on global polynomials, we use hierarchical basis functions (see [5, 10], with references therein). We then show how the hierarchical structure of the basis functions and the associated sparse grid can be used to refine the grid such that it can capture the local behavior of the functions to be interpolated (see, e.g. [28]). After explaining how this works, we provide examples showing that adaptive sparse grids outperform 'classical' sparse grids by far when it comes to interpolating functions that exhibit steep gradients or non-differentiabilities.

2.1 Notation

We first introduce some notation and definitions that we will require later [5, 10]. For all our considerations, we will focus on the domain $\Omega = [0,1]^d$, where d is the dimensionality of the problem. This situation can be achieved for other domains by a proper rescaling.
Let $\vec{l} = (l_1, ..., l_d) \in \mathbb{N}^d$ and $\vec{i} = (i_1, ..., i_d) \in \mathbb{N}^d$ denote multi-indices representing the grid refinement level as well as the spatial position of a d-dimensional grid point $\vec{x}_{\vec{l},\vec{i}}$. Using this notation, we can define the full grid $\Omega_{\vec{l}}$ on Ω with mesh size

$$h_{\vec{l}} := (h_{l_1}, ..., h_{l_d}) = 2^{-\vec{l}} := \left(2^{-l_1}, ..., 2^{-l_d}\right), \tag{1}$$

and gridpoints

$$\vec{x}_{\vec{l},\vec{i}} := (x_{l_1,i_1}, ..., x_{l_d,i_d}), \tag{2}$$

where $x_{l_t,i_t} := i_t \cdot h_{l_t} = i_t \cdot 2^{-l_t}$, and $i_t \in \{0, 1, ..., 2^{l_t}\}$. Along each dimension, the grid is equidistant. However, the mesh sizes, h_{l_t}, may differ across dimensions, in which case the grid is called anisotropic.
In addition, when dealing with d-dimensional multi-indices such as $\vec{l}$, we use relational operators component-wise,

$$\vec{l} \leq \vec{k} \Leftrightarrow l_t \leq k_t, \forall t \in \{1, \ldots, d\}. \tag{3}$$

Finally, we use the l_1-norm, $|\vec{l}|_1$, and the maximum norm, $|\vec{l}|_\infty$, given by

$$|\vec{l}|_1 := \sum_{t=1}^{d} l_t, \;\; |\vec{l}|_\infty := \max_{1 \leq t \leq d} l_t. \tag{4}$$

2.2 Hierarchical Basis Functions in One Dimension

We use a sparse grid method that is based on a hierarchical decomposition of the underlying approximation space. Such a hierarchical structure is crucial both for local adaptivity (see, Sec. 2.5) and for the use of parallel computing (see, Sec. 3.4). We now explain this hierarchical structure starting with the one dimensional case, i.e. $\Omega = [0, 1]$. Afterwards, we will extend it to the multivariate case using tensor products.

Let us assume that a function $f : \Omega \to \mathbb{R}$ of interest is sufficiently smooth [5]. For the time being we also assume that the function f vanishes at the boundary, i.e. $f|_{\partial\Omega} = 0$. We delegate the treatment of non-zero boundaries to Appendix A. An interpolation formula is then given by

$$f(x) \approx u(x) := \sum_i \alpha_i \phi_i(x) \tag{5}$$

with coefficients α_i and a set of appropriate piecewise linear basis functions $\phi_i(\vec{x})$. In the 'classical' sparse grid approach, standard hat functions

$$\phi(x) = \begin{cases} 1 - |x| & \text{if } x \in [-1, 1] \\ 0 & \text{else} \end{cases} \tag{6}$$

are used to generate a family of basis functions $\phi_{l,i}$ having support $[x_{l,i} - h_l, x_{l,i} + h_l]$ by dilation and translation, i.e.

$$\phi_{l,i}(x) := \phi\left(\frac{x - i \cdot h_l}{h_l}\right). \tag{7}$$

This basis is termed *nodal basis* [5]. The basis functions given in Eq. 7 are used to define the nodal function spaces

$$V_l := \text{span}\{\phi_{l,i} : 1 \leq i \leq 2^l - 1\}. \tag{8}$$

The hierarchical increment spaces W_l are defined by

$$W_l := \text{span}\{\phi_{l,i} : i \in I_l\}, \tag{9}$$

using the index set

$$I_l = \{i \in \mathbb{N}, 1 \leq i \leq 2^l - 1, i \text{ odd}\}. \tag{10}$$

In this way, the hierarchical increment spaces W_l are related to the nodal spaces V_l by the direct sum

$$V_l = \bigoplus_{k \leq l} W_k. \tag{11}$$

Fig. 1 shows the first three levels of these hierarchical, piecewise linear basis functions. Using this basis, a function f can be approximated by a unique $u \in V_l$ with coefficients $\alpha_{k,i} \in \mathbb{R}$:

$$f(x) \approx u(x) = \sum_{k=1}^{l} \sum_{i \in I_k} \alpha_{k,i} \cdot \phi_{k,i}(x). \tag{12}$$

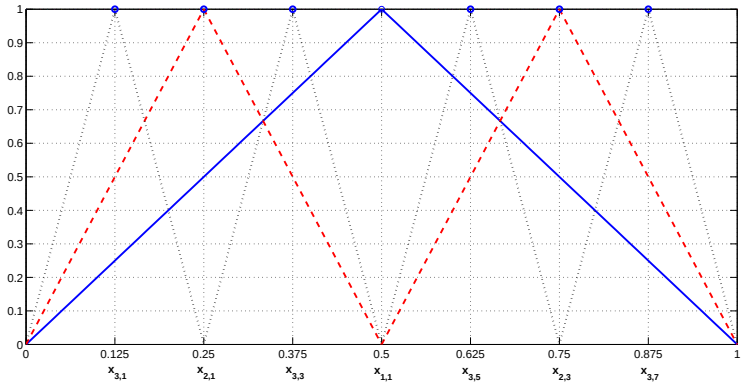

Figure 1: Hierarchical basis functions of V_3: level 1 (solid blue), level 2 (dashed red), level 3 (dotted black).

Note that the supports of all basis functions $\phi_{k,i}$ spanning W_k are mutually disjoint, as can be seen in Fig. 1. The coefficients $\alpha_{k,i}$ in the interpolant of the function f (see, Eq. 12) can easily be determined due to a nice property of the hierarchical grid, namely its nested structure: The set of points X^{l-1} at level $l-1$ with support nodes $x_{l,i}$ is contained in X^l, i.e. $X^{l-1} \subset X^l$.

In one dimension, the following relation for the hierarchical coefficients $\alpha_{l,i}$, $l \geq 1$, i odd holds [5, 10]:

$$\begin{aligned} \alpha_{l,i} &= f(x_{l,i}) - \frac{f(x_{l,i} - h_l) + f(x_{l,i} + h_l)}{2} \\ &= f(x_{l,i}) - \frac{f(x_{l,i-1}) + f(x_{l,i+1})}{2} \\ &= f(x_{l,i}) - \frac{f\left(x_{l-1,(i-1)/2}\right) + f\left(x_{l-1,(i+1)/2}\right)}{2}. \end{aligned} \tag{13}$$

In operator form, Eq. 13 can conveniently be rewritten as

$$\alpha_{l,i} = \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l,i} f, \tag{14}$$

Note that coefficients are called *hierarchical surpluses* [5] since a coefficient $\alpha_{l,i}$ corrects the interpolant of level $l-1$ at the points $x_{l,i}$ to the actual value of $f(x_{l,i})$, as displayed in Fig. 2.

2.3 Hierarchical Basis Functions in Multiple Dimensions

The one dimeional hierarchical basis can be extended to a d-dimensnsional one on the unit cube $\Omega = [0,1]^d$ by a tensor product construction. Our notation naturally extends to the d-dimensional case as well.

For each grid point $\vec{x}_{\vec{l},\vec{i}}$, an associated piecewise d-linear basis function $\phi_{\vec{l},\vec{i}}(\vec{x})$ is defined as the product of the one-dimensional basis functions (see, Eq. 6)

$$\phi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{t=1}^{d} \phi_{l_t,i_t}(x_t). \tag{15}$$

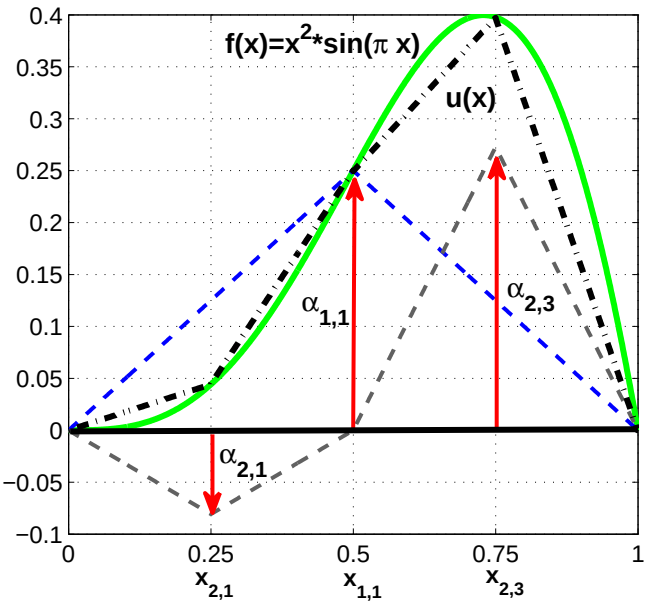

Figure 2: Construction of $u\,(x)$ interpolating $f\,(x) = x^2 \cdot \sin(\pi \cdot x)$ with hierarchical linear basis functions of levels 1 and 2. The *hierarchical surpluses* $\alpha_{l,i}$ that belong to the respective basis functions are indicated by arrows (cf., Eq. 13). They are simply the difference between the function values at the current and the previous interpolation levels.

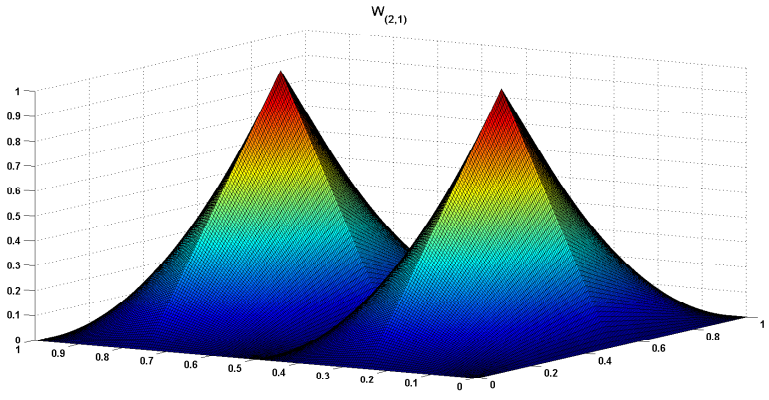

Figure 3: Basis functions of the hierarchical increment space $W_{(2,1)}$.

These basis functions are then used to define the function spaces $V_{\vec{l}}$ consisting of piecewise linear functions on Ω with $f|_{\partial\Omega} = 0$:

$$V_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{1} \leq \vec{i} \leq 2^{\vec{l}} - \vec{1}\}. \tag{16}$$

Again, the index set $I_{\vec{l}}$ is given by

$$I_{\vec{l}} := \{\vec{i} : 1 \leq i_t \leq 2^{l_t} - 1, i_t \text{ odd}, 1 \leq t \leq d\}. \tag{17}$$

Note that the hierarchical increments, formally defined as

$$W_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}, \tag{18}$$

can alternatively be written as

$$W_{\vec{l}} := V_{\vec{l}} \setminus \bigoplus_{t=1}^{d} V_{\vec{l}-\vec{e}_t}, \tag{19}$$

where $\vec{e}_t$ is the t-th unit vector. In other words, $W_{\vec{l}}$ consist of all $\phi_{\vec{l},\vec{i}} \in V_{\vec{l}}$ (using the hierarchical basis functions) which are not included in any of the spaces $V_{\vec{k}}$ smaller than $V_{\vec{l}}$.[1] An example of such a function space is given in Fig. 3. These hierarchical difference spaces now allow us to define a multilevel space decomposition. In line with the sparse grid literature (see, e.g. [28, 10, 5]), we define $V_n := V_{\vec{n}}$ as a direct sum of spaces. Consequently, the hierarchical increment spaces $W_{\vec{l}}$ are related to the nodal spaces $V_{\vec{l}}$ of piecewise d-linear functions with mesh width h_l in each dimension by

$$V_n := \bigoplus_{l_1=1}^{n} \cdots \bigoplus_{l_d=1}^{n} W_{\vec{l}} = \bigoplus_{|l|_\infty \leq n} W_{\vec{l}}, \tag{20}$$

leading to a full grid with $(2^n - 1)^d$ gridpoints. The interpolant of f, namely $u(\vec{x}) \in V_n$, can uniquely be represented by

$$f(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_\infty \leq n} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|l|_\infty \leq n} f_{\vec{l}}(\vec{x}), \tag{21}$$

with $f_l \in W_{\vec{l}}$ and $\alpha_{\vec{l},\vec{i}} \in \mathbb{R}$. In the d-dimensional case, the hierarchical surpluses are given by

$$\alpha_{\vec{l},\vec{i}} = \left(\prod_{t=1}^{d} \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l_t, i_t} \right) f. \tag{22}$$

For a sufficiently smooth function f (which we will make precise in the next section) and its interpolant $u \in V_n$ [5], we obtain an asymptotic error decay of

$$\| f(\vec{x}) - u(\vec{x}) \|_{L_2} \in \mathcal{O}\left(h_n^2\right), \tag{23}$$

but at the cost of

$$\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right) \tag{24}$$

function evaluations, encountering the so-called *curse of dimensionality*. The exponential dependence of the overall computational effort on the number of dimensions is a prohibitive obstacle for the numerical treatment of high-dimensional problems. The curse of dimensionality typically prohibits an accurate solution of problems with more than four or five dimensions. For example a resolution of 15 points in each dimension, i.e. $n = 4$, for a ten-dimensional problem needs $0.58 \cdot 10^{12}$ coefficients, which already brings us to the capacity limits of today's most advanced computer systems [8].

[1] Note that a function space $V_{\vec{k}}$ is called 'smaller' than a space $V_{\vec{l}}$ if $\forall k_t \leq l_t$ and $\exists t : k_t < l_t$. In this case, the grid $\Omega_{\vec{k}}$ is also called smaller than the grid $\Omega_{\vec{l}}$.

2.4 Classical Sparse Grids

As a consequence of the curse of dimensionality (see, Sec. 2.3), the question that needs to be answered is how we can construct discrete approximation spaces that are better than V_n in the sense that the same number of invested gridpoints leads to a higher order of accuracy [39, 5]. The 'classical' sparse grid construction arises from a 'cost-to-benefit' analysis (see, e.g., [39, 10, 5], with references therein) in function approximation. Thereby, functions $f(\vec{x}) : \Omega \to \mathbb{R}$ which have bounded mixed derivatives,

$$D^{\vec{l}} f := \frac{\partial^{|\vec{l}|_1}}{\partial x_1^{l_1} \cdots \partial x_d^{l_d}} f \tag{25}$$

for $|l|_\infty \leq 2$ are considered. These functions belong to a Sobolev space

$$H_2^{\text{mix}}(\Omega) := \{f : \Omega \to \mathbb{R} : D^{\vec{l}} f \in L_2(\Omega), |\vec{l}|_\infty \leq 2, f|_{\partial\Omega} = 0\}. \tag{26}$$

For functions in $H_2^{\text{mix}}(\Omega)$ the hierarchical coefficients $\alpha_{\vec{l},\vec{i}}$ (see, Eq. 21 and [5]) rapidly decay for functions $f \in H_2^{\text{mix}}$, namely

$$|\alpha_{\vec{l},\vec{i}}| = \mathcal{O}\left(2^{-2|\vec{l}|_1}\right). \tag{27}$$

The strategy for constructing a sparse grid is to leave out those subspaces among the full grid space V_n that only contribute little to the interpolant [5]. An optimization with respect to the number of degrees of freedom, i.e. the gridpoints, and the resulting approximation accuracy directly leads to the *sparse grid* space $V_{0,n}^S$ of level n, defined by

$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}, \tag{28}$$

where the index 0 in $V_{0,n}^S$ stands for $f|_{\partial\Omega} = 0$. Note that the concrete choice of subspaces depends on the norm in which we measure the error. The result obtained in Eq. 28 is optimal for the L_2-norm and the L_∞-norm [5].
The number of gridpoints required by the space $V_{0,n}^S$ is now given by [5, 10]

$$|V_{0,n}^S| = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}\left(n^{d-2}\right)\right) = \mathcal{O}\left(h_n^{-1} \cdot \left(\log(h_n^{-1})\right)^{d-1}\right). \tag{29}$$

This is of order $\mathcal{O}\left(2^n \cdot n^{d-1}\right)$, which is a significant reduction of the number of gridpoints, and thus of the computational and storage requirements compared to $\mathcal{O}\left(2^{nd}\right)$ of the full grid space $|V_n|$ (see, Tabs. 1 and 8). In analogy to Eq. 21, a function $f \in V_{0,n}^S \subset V_n$ can now be expanded by

$$f_{0,n}^S(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|l|_1 \leq n+d-1} f_{\vec{l}}(\vec{x}), \tag{30}$$

where $f_{\vec{l}} \in W_{\vec{l}}$. Note that $\alpha_{\vec{l},\vec{i}} \in \mathbb{R}$ are commonly termed the *hierarchical surpluses* [39, 5]. They are simply the difference between the function values at the current and the previous interpolation levels (see, Fig. 2). The nice

Dimension d	Full Grid $\lvert V_4 \rvert$	Sparse Grid $\lvert V_{0,4}^S \rvert$
1	15	15
2	225	49
3	3'375	111
4	50'625	209
5	759'375	351
10	$5.77 \cdot 10^{11}$	2'001
15	$4.37 \cdot 10^{17}$	5'951
20	$3.33 \cdot 10^{23}$	13'201
30	$1.92 \cdot 10^{35}$	41'601
40	$1.11 \cdot 10^{47}$	95'201
50	$6.38 \cdot 10^{58}$	182'001
100	>Googol	1'394'001

Table 1: Number of gridpoints for increasing dimension and two different types of grids of refinement level 4.

thing about Eq. 30 is that it allows to utilize the results generated previously to improve the interpolation. As we have chosen our set of gridpoints to be nested, i.e. such that the set of points X^{l-1} at level $l-1$ with support nodes $\vec{x}_{\vec{l},\vec{i}}$ is contained in X^l, namely $X^{l-1} \subset X^l$, the extension of the interpolation level from level $l-1$ to l only requires to evaluate the function at gridpoints that are unique to X^l, that is, at $X_\Delta^l = X^l \backslash X^{l-1}$.

The asymptotic accuracy of the interpolant deteriorates only slightly from $\mathcal{O}\left(h_n^2\right)$ in case of the full grid (cf., Eq. 23) down to

$$\mathcal{O}\left(h_n^2 \cdot \log(h_n^{-1})^{d-1})\right), \tag{31}$$

as shown in [5, 10]. Taken together, Eqs. 29 and 31 demonstrate why sparse grids are so well-suited for high-dimensional problems. In contrast to full grids, their size increases only moderately with dimension, while the accuracy they provide is only slightly worse than of full grids.

Note that sparse grid methods are not restricted to piecewise linear basis functions; there are several other basis functions possible, e.g. piecewise d-polynomial ones (see, [5, 28], with references therein). However, we focus on simple linear hat functions as they have non-overlapping support, which is convenient for adaptive refinement procedures as presented below in Sec. 2.5.
Finally, note that Eq. 31 also holds for sparse grids with non-vanishing boundaries, i.e. $f|_{\partial\Omega} \neq 0$ [5]. For sparse grid constructions with non-zero boundaries, we point the reader to Appendix A.

2.5 Adaptive Sparse Grids

The sparse grid structure introduced in the previous sections defines an a priori selection of gridpoints that is optimal if certain smoothness conditions are met, i.e. if the function has bounded mixed derivatives (cf., Sec. 2.4 and Eq. 26). However, in many applications (see, e.g. [28], with references therein) including economic models with occasionally binding constraints, these prerequisites are not met: the functions of interest often show kinks or finite discontinuities.

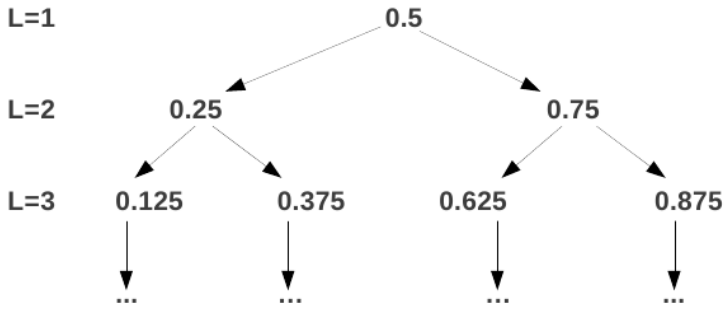

Figure 4: One-dimensional tree-like structure of a 'classical' sparse grid (cf., Sec. 2.4) for the first three hierarchical levels.

Thus, the sparse grid methods outlined so far may fail to provide good approximations, as they can capture the local behaviour only to some limited extend. Therefore, the next task is to find a way of efficiently approximating functions which do not fulfill the necessary smoothness conditions given in Eq. 26.

A very effective strategy to achieve this is to adaptively refine the sparse grid at points near steep or non-differentiable regions and spend less points in regions of low function variation (see, e.g. [29, 28, 13, 4, 24]). By doing so, resources are only invested where needed. While there are various ways to refine a sparse grid (see, e.g. [28], with references therein), we outline only briefly the basic ideas behind the algorithms that we are using in the course of solving our economic models below and omit the technical details. For these, we refer the reader to the original articles, namely the ones by [24] and [28].

When approximating a function as a sum of piecewise linear basis functions, we can hope that the main contributions to the interpolant stem from comparatively few terms with big surpluses (cf., Eq. 30 and Fig. 2). The key point of the refinement strategies of [24, 28] therefore is to monitor the size of the hierarchical surpluses. Recall from Sec. 2.3 and Sec. 2.4 that the interpolated function is represented by a linear combination of hierarchical, piecewise linear hat functions. The coefficients of the hat functions - the hierarchical surpluses - are just the hierarchical increments between two successive interpolation levels. The magnitude of the hierarchical surplus reflects the local irregularity of the function. For smooth functions, its value tends to zero as the level l tends to infinity (cf., Eq. 27). On the other hand, for a non-smooth function, a singularity/discontinuity is indicated by the magnitude of the hierarchical surplus. Therefore, the hierarchical surplus serves as a natural error indicator.

Technically, the adaptive grid refinement can be built on top of the hierarchical grid structure. Let us first consider the one dimensional case. The equidistant gridpoints form a tree-like data structure [24], as displayed in Fig. 4. Going from one level to the next, we see that for each grid point there are two *sons*. For example, the point 0.5 from level $l = 1$ is the *father* of the points 0.25 and 0.75 from level $l = 2$. In the d-dimensional case, there are consequently two *sons* in each dimension for each grid point, i.e. $2d$ *sons*, when going from one to the next hierarchical grid level. Moreover, note that the *sons* are also the neighbor points of the *father*. Recall now from Sec. 2.4 and Eq. 17 that the

neighboring points are the support nodes of the hierarchical basis in the next interpolation level. Therefore, by adding neighboring points, we simply add the support nodes from the next interpolation level, i.e. we refine an interpolation from level $l-1$ to l. In order to adaptively refine the grid, we use the hierarchical surpluses as an error indicator in order to detect the smoothness of the solution and refine those hierarchical basis functions $\phi_{\vec{l},\vec{i}}$ which have a hierarchical surplus, $\alpha_{\vec{l},\vec{i}}$, that satisfies

$$|\alpha_{\vec{l},\vec{i}}| \geq \epsilon, \tag{32}$$

for a so-called refinement threshold $\epsilon \geq 0$. Whenever this criterion is satisfied, $2d$ neighbor points of the current point are added to the sparse grid. Note that this refinement method scales linearly and thus does not suffer from the curse of dimensionality. For more technical information regarding the implementation of adaptive sparse grid methods, we refer to [24, 28].
Note that in our application in Sec. 3, we interpolate several policies on one grid, i.e. we interpolate a function

$$f : \Omega \to \mathbb{R}^m.$$

Therefore, we get m surpluses at each gridpoint and we thus have to replace the refinement criterion in Eq. 32 by

$$g\left(\alpha^1_{\vec{l},\vec{i}}, \ldots, \alpha^m_{\vec{l},\vec{i}}\right) \geq \epsilon, \tag{33}$$

where the refinement choice is governed by a function $g : \mathbb{R}^m \to \mathbb{R}$. A natural choice for g is the maximum function, which we will use in Sec. 3.5.

Analytical Examples

We now demonstrate the ability of adaptive sparse grid algorithms to efficiently interpolate functions that exhibit steep gradients and kinks. This part contains analytical tests in one and two dimensions in order to foster the understanding of the adaptive sparse grid algorithms in use, namely the ones by [28] and [24]. Their behaviour will be investigated with respect to different grid structures. Note, however, that these algorithms were extensively and carefully tested before, e.g. via the test problems by [11]. Therefore, we restrict ourselves below to a few examples.

For the testing, we proceed as follows: We pick a (non-smooth) function $f : [0,1] \to \mathbb{R}$, construct the interpolant $u(\vec{x})$ of $f(\vec{x})$ (cf., Eq. 30), then randomly generate 1000 test points from a uniform distribution in $[0,1]^d$, and finally compute the maximum error given by

$$\max_{i=1,\ldots,1000} |f(\vec{x_i}) - u(\vec{x_i})|. \tag{34}$$

Moreover, we also assess which choice of the sparse grid and its respective basis functions suits our purposes best, either the one by [28] or the one by [24]. More precisely, we compare the following two settings:

- *setting A* (algorithm by [24]):
  - grid: Curtis-Clenshaw grid (cf., Eq. 63)

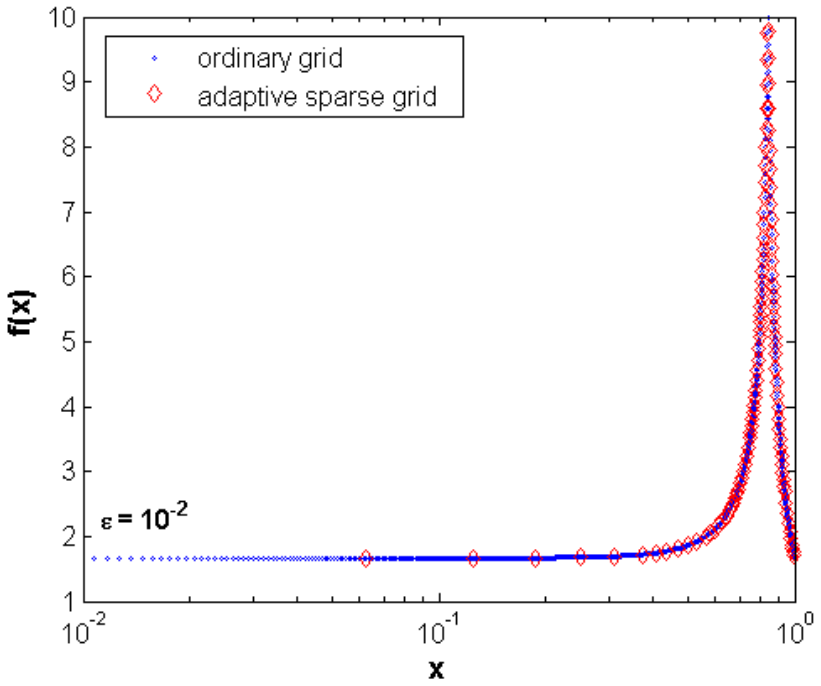

Figure 5: Evaluation of the function given by Eq. 35 at the points of the adaptive sparse grid (red diamonds) and the 'full grid' (blue dots). Both grids attain a maximum error $\mathcal{O}\left(10^{-2}\right)$. The adaptive sparse grid reaches this level of convergence with 109 points, whereas the full grid needs 1023 points.

  - basis functions: modified linear basis functions (cf., Eq. 64)

- *setting B* (algorithm by [28]):
  - grid: 'standard' sparse grid (cf., Eq. 10)
  - basis functions: modified linear basis functions (cf., Eq. 65)

As a first educational example, we apply *'setting B'* to the one dimensional test function

$$f(x) = \frac{1}{|0.5 - x^4| + 0.01}. \tag{35}$$

The refinement threshold for the adaptive sparse grid algorithm (cf., Eq. 32) is chosen to be $\epsilon = 10^{-2}$. With this setting, the maximum interpolation error reaches $\mathcal{O}\left(10^{-2}\right)$ (cf., Eq. 34), while 109 gridpoints have to be spent. In contrast, to attain the same level of accuracy, 1023 equidistant gridpoints have to be spent,[2] as shown in Fig. 5. From Fig. 5, it is obvious that the adaptive sparse grid places points in regions where high resolution is needed, while putting only few points in areas where the function varies little. This fact makes adaptive

[2]Note that in the one dimensional case, an ordinary sparse grid of level l corresponds to the full grid with the same level of refinement.

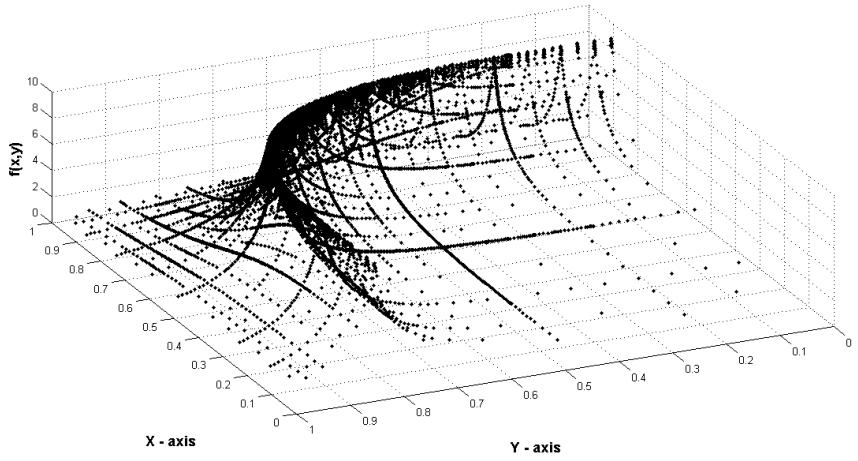

Figure 6: Evaluation of Eq. 36 at the gridpoints obtained by the adaptive sparse grid algorithm 'setting B' after 15 refinement steps.

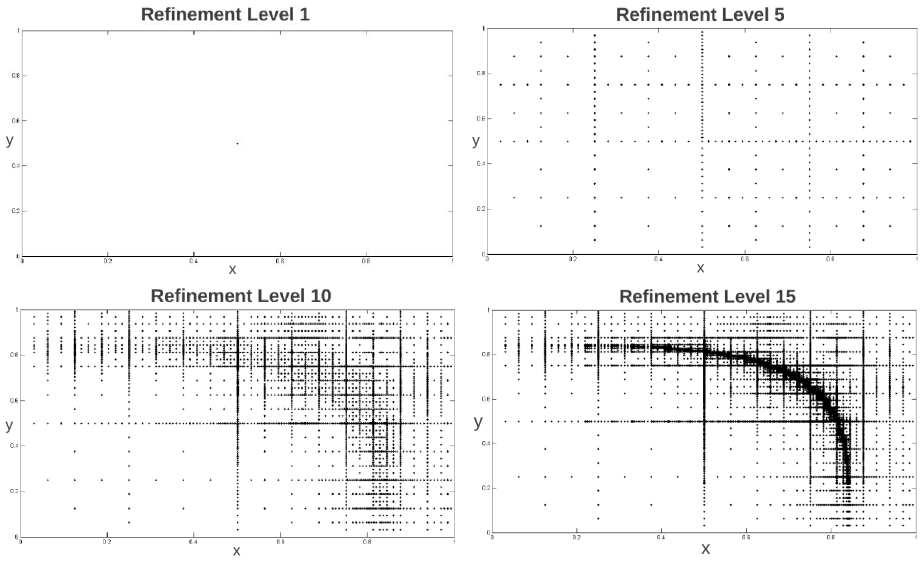

Figure 7: The evolution of an adaptive sparse grid on $[0,1]^2$ with a threshold $\epsilon = 10^{-2}$. The refinement levels $1, 5, 10$ and 15 are shown.

sparse grid algorithms favourable over all other (sparse) grid interpolation methods if kinks or discontinuities have to be handled. As non-adaptive methods can only provide one resolution over the whole domain, they waste resources where not needed.

As a second example, we apply both 'setting A' and 'setting B' with a

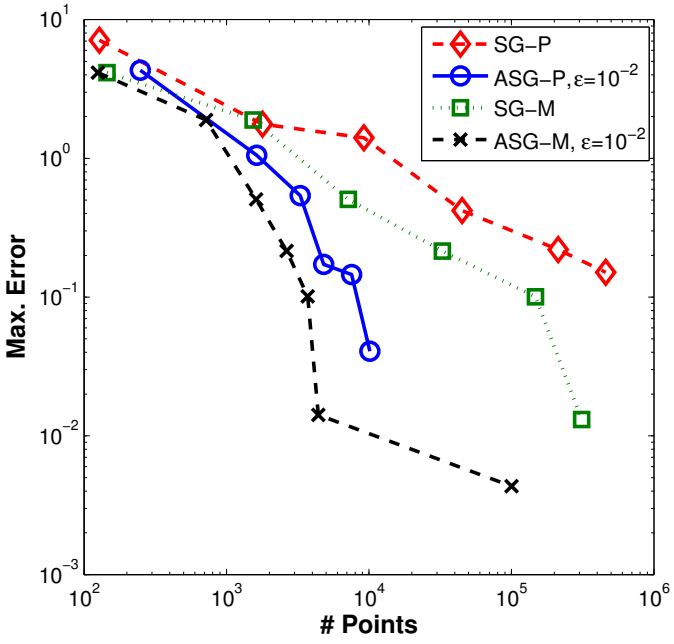

Figure 8: Comparison of the interpolation error (cf., Eq. 34) for conventional and adaptive sparse grid interpolation at different refinement levels. Note that the adaptive sparse grid algorithm 'setting A' is labelled by 'ASG-M', while its corresponding 'classical' sparse grid version is denoted by 'SG-M'. In analogy, 'setting B' is denoted by 'ASG-P' (and 'SG-P'). Both adaptive grids were obtained by applying a threshold $\epsilon = 10^{-2}$.

threshold of $\epsilon = 10^{-2}$ to the two dimensional test function

$$f(x, y) = \frac{1}{|0.5 - x^4 - y^4| + 0.1}, \tag{36}$$

a line-singularity, as shown in Fig. 6.

In Fig. 8, we provide the convergence rate of the adaptive sparse grid method for 'setting A' and 'setting B'. The data points shown in Fig. 8 were obtained by computing the errors (cf., Eq. 34) at the refinement steps 5, 8, 10, 12 and 15. These results are contrasted by 'classical' sparse grid counterparts of the respective refinement level.
Strikingly, 'setting A' for example reaches an accuracy of $e \approx 1.4 \cdot 10^{-2}$, where the interpolation refinement level is 15 and the number of points is $4'411$ as opposed to $311'297$ points using the same level of refinement in the conventional sparse grid. An exemplary evolution of the adaptive sparse grid is shown in Fig. 7. Note that the line of discontinuity is automatically detected by the adaptive sparse grid algorithm (cf., Figs. 7 and 6). From Fig. 8, it also gets apparent that 'setting A' is generally converging even faster than 'setting B'. Thus, we will use 'setting A' to solve the international real business cycle model in Section 3.

3 Application to Dynamic Models

In this section, we apply the adaptive sparse grid method to an economic example, namely a international real business cycle model. We first introduce the model in Sec. 3.1, then present an extension with irreversible investment in Sec. 3.2, and subsequently outline the time iteration algorithm in Sec. 3.3. Finally, we present the parallelization of the code in Sec. 3.4 and discuss its performance in Sec. 3.5.

3.1 International Real Business Cycle Model

To demonstrate the capabilities of adaptive sparse grids in solving dynamic economic models we apply this method to a real business cycle model with multiple countries, i.e. an international real business cycle model. This model has become a standard for testing computational methods for solving high-dimensional dynamic models (see, [7], with references therein).

Model Description

There are N countries that differ with respect to their exogenous productivity (and possibly preferences) as well as their endogenous capital stock. They all produce, trade and consume a single homogeneous good. Production of country j at time t is given by

$$y_t^j = a_t^j \cdot f^j(k_t^j) \tag{37}$$

where a_t^j, f^j, and k_t^j are productivity, a neoclassical production function, and the capital stock of country j respectively. The law of motion of productivity is given by

$$\ln a_t^j = \ln a_{t-1}^j + \sigma \left(e_t^j + e_t \right), \tag{38}$$

where the shock e_t^j is specific to country j, while e_t is a global shock. These shocks are all i.i.d. standard normal.

The law of motion of capital is given by

$$k_{t+1}^j = k_t^j \cdot (1 - \delta) + i_t^j, \tag{39}$$

where δ is the rate of capital depreciation, and i_t^j is investment. There is a convex adjustment cost on capital, given by

$$\Gamma_t^j(k_t^j, k_{t+1}^j) = \frac{\phi}{2} \cdot k_t^j \cdot \left(\frac{k_{t+1}^j}{k_t^j} - 1 \right)^2. \tag{40}$$

The aggregate (i.e. global) resource constraint is thus given by

$$\sum_{j=1}^{N} y_t^j \geq \sum_{j=1}^{N} \left(i_t^j + \Gamma_t^j(k_t^j, k_{t+1}^j) + c_t^j \right), \tag{41}$$

where c_t^j denotes consumption of country j at time t. Substituting and rearranging, we get

$$\sum_{j=1}^{N} \left(a_t^j \cdot f^j(k_t^j) + k_t^j \cdot (1 - \delta) - k_{t+1}^j - \Gamma_t^j(k_t^j, k_{t+1}^j) - c_t^j \right) \geq 0. \tag{42}$$

We assume that the preferences of each country are represented by a time separable utility function with discount factor β and per-period utility function u^j. By further assuming complete markets, the decentralized competitive equilibrium allocation can be obtained as the solution to a social planner's problem, where the welfare weights, τ^j, of the various countries depend on their initial endowments. More precisely, the social planner solves

$$\max_{\{c_t^j, k_t^j\}} \mathbb{E}_0 \sum_{j=1}^{N} \tau^j \cdot \left(\sum_{t=1}^{\infty} \beta^t \cdot u^j(c_t^j) \right), \tag{43}$$

subject to the aggregate resource constraint (42).

First Order Conditions

To get the first order conditions (FOCs) of problem (43), we differentiate the Lagrangian with respect to c_t^j

$$\tau^j \cdot u_c^j(c_t^j) - \lambda_t = 0, \tag{44}$$

and with respect to k_{t+1}^j:

$$\begin{aligned} &\lambda_t \left[-1 - \frac{\partial \Gamma_t^j(k_t^j, k_{t+1}^j)}{\partial k_{t+1}^j} \right] + \\ &\quad \beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \cdot \left[a_{t+1}^j \cdot f_k^j(k_{t+1}^j) + (1-\delta) - \frac{\partial \Gamma_{t+1}^j(k_{t+1}^j, k_{t+2}^j)}{\partial k_{t+1}^j} \right] \right\} = 0, \end{aligned} \tag{45}$$

where λ_t denotes the multiplier on the time t resource constraint, and derivatives are denoted like $u_c(c) = \partial u(c)/\partial c$. Differentiating the adjustment cost function given in Eq. 40, simplifying, and defining the growth rate of capital by $g_t^j = k_t^j/k_{t-1}^j - 1$, Eq. 45 reads:

$$\begin{aligned} &-\lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j \right] + \\ &\quad \beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \cdot \left[a_{t+1}^j \cdot f_k^j(k_{t+1}^j) + 1 - \delta + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2 \right) \right] \right\} = 0. \end{aligned} \tag{46}$$

Concerning the production function $f^j(k_t^j)$ and the marginal utility function $u_c^j(c_t^j)$, we assume the following standard functional forms:

$$f^j(k_t^j) = A \cdot (k_t^j)^{\alpha} \tag{47}$$

and

$$u_c^j(c_t^j) = (c_t^j)^{-\frac{1}{\gamma_j}}, \tag{48}$$

which imply

$$f_k^j(k_t^j) = A \cdot \alpha \cdot (k_t^j)^{\alpha - 1}, \tag{49}$$

$$c_t^j = \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \tag{50}$$

Parameter	Symbol	Value
discount factor	β	0.99
IES of country j	γ^j	a+(j-1)(b-a)/(N-1) with a=0.25, b=1
capital share	α	0.36
depreciation	δ	0.01
std. of log-productivity shocks	σ	0.01
autocorrelation of log-productivity	ρ	0.95
intensity of capital adjustment costs	ϕ	0.50
number of countries	N	2, 4, 6, 8, 10, 11, or 12

Table 2: Choice of parameters for the IRBC model

Using these expressions, we finally get the system of $N+1$ equilibrium conditions that we solve in our computations, namely for all countries $j \in \{1, ..., N\}$:

$$\begin{aligned} &\lambda_t \cdot \left[1 + \phi \cdot g^j_{t+1}\right] - \\ &\beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \left[a^j_{t+1} \cdot A \cdot \alpha \cdot (k^j_{t+1})^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g^j_{t+2} \cdot \left(g^j_{t+2} + 2 \right) \right] \right\} = 0, \end{aligned} \tag{51}$$

and the aggregate resource constraint

$$\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^{\alpha} + k^j_t \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g^j_{t+1})^2 \right) - k^j_{t+1} - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0. \tag{52}$$

We solve the IRBC model by iterating on Eqs. 51 and 52. The implementation details are provided in Sec. 3.3.

Parameterization

With respect to the parameter choices, we follow Juillard and Villemot [19], who provide the model specifications for the comparison study that uses the IRBC model to test several solution methods (see, [7]). We choose an asymmetric specification where preferences are heterogeneous across countries. In particular, the intertemporal elasticity of substitution (IES) of the N countries is evenly spread over the interval $[0.25, 1]$. This corresponds to model A5 in Juillard and Villemot [19]. The only difference between our parameterization and theirs is that we use a (quarterly) depreciation rate of $\delta = 1\%$, while they write down the model such that they have effectively no depreciation. The parameters we use are reported in Tab. 2. The welfare weights τ^j need not to be specified as they do not matter for the capital allocation, but only for the consumption allocation which we do not consider. The parameter A is chosen such that capital of each country is equal to 1 in the deterministic steady state.

3.2 IRBC Model With Irreversible Investment

To demonstrate that our algorithm can handle non-smooth behavior, we include irreversible investment in the IRBC model of Sec. 3.1. More precisely, we assume

that investment cannot be negative, thus for each country $j \in \{1, ..., N\}$ the following constraint has to be satisfied:

$$k_{t+1}^j \geq k_t^j \cdot (1 - \delta). \tag{53}$$

As a consequence, we have to solve a system of $2N + 1$ equilibrium conditions. These conditions now include the Kuhn-Tucker multiplier for the irreversibility constraint $k_{t+1}^j - k_t^j \cdot (1 - \delta) \geq 0$, which we denote by μ_t^j. The Euler equations and the irreversibility constraints for all countries $j \in \{1, ..., N\}$ are given by:

$$\begin{aligned}
&\lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j\right] - \mu_t^j \\
&- \beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \left[a_{t+1}^j \cdot A \cdot \alpha \cdot (k_{t+1}^j)^{\alpha - 1} + (1 - \delta) + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2 \right) \right] \right\} = 0, \\
&k_{t+1}^j - k_t^j (1 - \delta) \geq 0, \mu_t^j \geq 0, \left(k_{t+1}^j - k_t^j (1 - \delta) \right) \cdot \mu_t^j = 0
\end{aligned} \tag{54}$$

and also the aggregate resource constraint:

$$\sum_{j=1}^{N} \left(a_t^j \cdot A \cdot (k_t^j)^{\alpha} + k_t^j \cdot \left((1 - \delta) - \frac{\phi}{2} \cdot (g_{t+1}^j)^2 \right) - k_{t+1}^j - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0. \tag{55}$$

The parameters we use for the IRBC model with irreversible investment are the same as the ones we use for the IRBC model.

3.3 Time Iteration Algorithm

To compute an equilibrium of the IRBC model, we embed adaptive sparse grid interpolation in a time iteration algorithm (see, e.g. [18]).[3] We solve for an equilibrium that is Markov in the physical state of the economy, which is given by the capital stock and the productivity levels of all N countries:

$$\left(a_t^1, \ldots, a_t^N, k_t^1, \ldots, k_t^N \right). \tag{56}$$

Denoting the state space by $S \subset \mathbb{R}_+^{2N}$, we thus solve for policies

$$p = \left(k_{t+1}^1, \ldots, k_{t+1}^N, \lambda_t \right) : S \to \mathbb{R}_+^{N+1}. \tag{57}$$

Such policies represent an equilibrium of the IRBC model, only if they satisfy Eqs. 51 and 52 at all points in the state space. To compute policies that approximately satisfy this condition, we use time iteration and employ adaptive sparse grid interpolation in each of its iteration steps. The structure of the algorithm is as follows:[4]

1. Make an initial guess for next period's policy function:

$$p_{init} = \left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1} \right).$$

   Set $p_{next} = p_{init}$. Choose an approximation accuracy $\bar{\eta}$.

[3] Note that we present the time iteration algorithm for the smooth IRBC model – however, the time-iteration procedure for the non-smooth IRBC works analogously—Eqs. 51 and 52 just need to be replaced by Eqs. 54 and 55.

[4] This is the algorithm for the adaptive sparse grid. The 'classical' sparse grid of level L is obtained as a special case by setting $L_0 = 1$, $\epsilon = 0$ and $L_{max} = L$.

2. Make one time iteration step:

   (a) Start with a coarse grid $G_{old} \subset S$ (a 'classical' sparse grid of a low level L_0), and generate G by adding for each $g \in G_{old}$ all $2d$ neighbouring points. Choose a maximal refinement level $L_{max} > L_0$ and set $l = 1$.

   (b) For all
   $$g = \left(a_t^1, \ldots, a_t^N, k_t^1, \ldots, k_t^N\right) \in G$$
   solve[5] for the optimal policies
   $$p(g) = \left(k_{t+1}^1(g), \ldots, k_{t+1}^N(g), \lambda_t(g)\right)$$
   by solving the system of equilibrium conditions (Eqs. 51 and 52) given next period's policy
   $$p_{next} = \left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1}\right).$$

   (c) Generate G_{new} from G by adding for each $g \in G_l \setminus G_{old}$ its $2d$ neighbouring points, if
   $$\|p(g) - \tilde{p}(g)\|_\infty > \epsilon$$
   where the policy $\tilde{p}(g)$ is given by interpolating between $\{p(g)\}_{g \in G_{old}}$.

   (d) If $G_{new} = G$ or $L_0 + l = L_{max}$, then set $G = G_{new}$ and go to (e), else set $l = l + 1$ and go to (b).

   (e) Define the policy function p by interpolating between $\{p(g)\}_{g \in G}$.

   (f) Calculate (an approximation for) the error, e.g.
   $$\eta = \|p - p_{next}\|_\infty.$$
   If $\eta > \bar{\eta}$, set $p_{next} = p$ and go to step 2, else go to step 3.

3. The (approximate) equilibrium policy function is given by p.

In principle, one could set the maximum refinement level L_{max} to a very large value such that it is never reached for a given refinement threshold. However, this can create practical problems as one has no reasonable upper bound for the number of gridpoints created by the refinement procedure.[6]

3.4 Parallelization and Scaling

In order to enable the solution of 'large' problems in a reasonable amount of time, we aim to access modern high-performance computing architectures. The time iteration algorithm used to solve the IRBC model is therefore parallelized by a hybrid parallelization scheme, i.e. with MPI [34] (distributed memory parallelization) between the compute nodes and OpenMP [16] (shared memory parallelization) within the nodes. We achieve this as follows. We construct adaptive sparse grids according to the algorithm of [24] (cf., 'setting A', Sec.

[5] Of course, if $l > 1$ one only has to solve for all $g \in G \setminus G_{old}$.

[6] Note that for the 4-dimensional models (see, Sec. 3.5), we set L_{max} such that it is never reached. However, for the solution of higher-dimensional non-smooth IRBC models, we set $L_{max} = 6$ (see, Tab. 7).

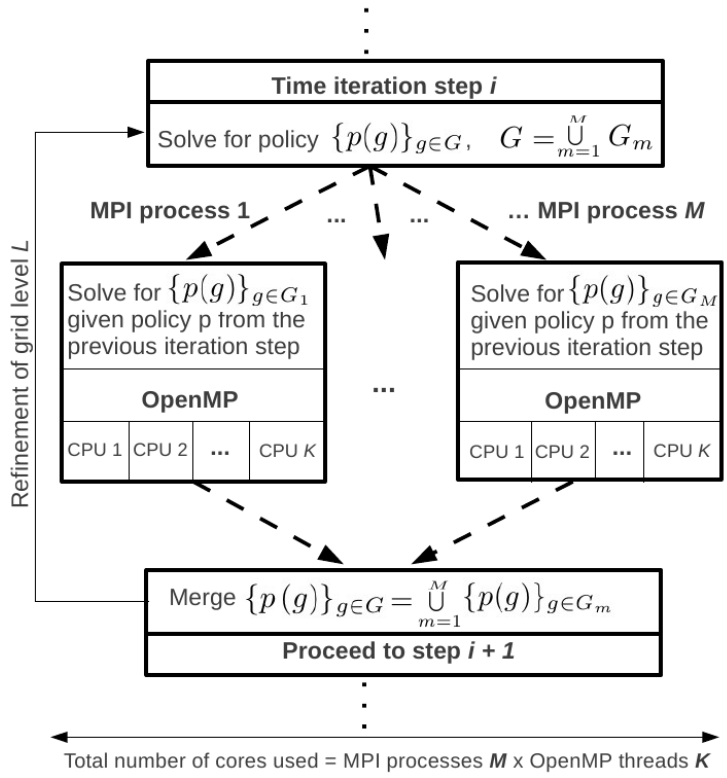

Figure 9: Schematic representation of the hybrid parallelization of a time iteration step (see, Sec. 3.3). The total number of cores used is given by the number of MPI processes times the number of OpenMP threads.

2.5), which allows for an MPI parallel evaluation of the hierarchical surpluses, i.e. it distributes the newly generated points within a refinement step among different multicores (see, Fig. 9). On top of this, we add an additional level of parallelism. Locally, we solve the nonlinear system of equations (see, Sec. 3.1 and Eqs. 51 and 52 or Eqs. 54 and 55, respectively) in a shared memory fashion (OpenMP) in order to evaluate the hierarchical surpluses at these particular gridpoints. A schematic illustration of one time iteration step (point 2. in the above algorithm) is displayed in Fig. 9. Note that in our implementation, we solve the set of nonlinear equations with IPOPT [37] in combination with PARDISO [33, 32]. The advantage of this parallelization scheme is that it drastically reduces the amount of MPI communication required between different multicores. This fact is important, as the MPI communication overhead between different processes can turn out to be a roadblock for the efficiency of the code when going to 'large' process numbers [34].

Indicative performance numbers are shown in Fig. 10, where we display the speedup S_p and the efficiency E_p of the code. These two quantities are defined by [31]

$$S_p = \frac{T_1}{T_p}, \qquad E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}, \tag{58}$$

where T_1 is the execution time of the test benchmark, T_p is the execution time of the algorithm running with a multiple of p-times the processes of the baseline. We see that the code scales nicely to at least about $1,200$ parallel processes, as shown in Fig. 10.

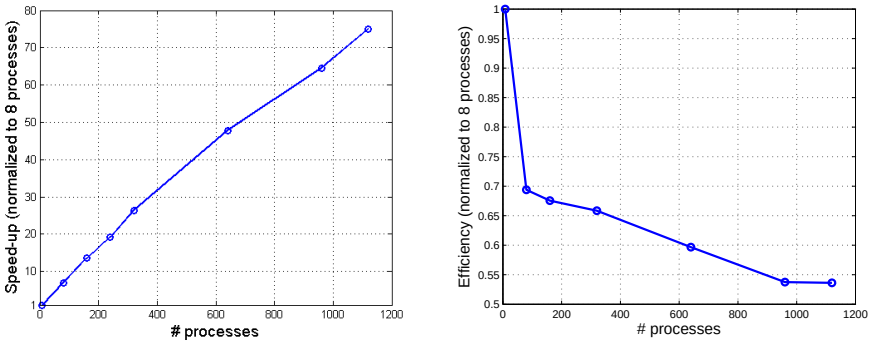

Figure 10: The figure shows 'strong scaling' of the code under a hybrid parallelization with MPI between nodes and Open MP within nodes. The test was performed on the Schrödinger system (nodes with 8 core Intel Xeon X5560 2.8 GHz processors) at the University of Zürich and consisted of one representative time step. The test problem was a 10-dimensional 'classical' sparse grid of refinement level 3 on which we solve Eqs. 51 and 52. The speedup, normalized to 8 processes, is shown on the left hand side and refers to how much the parallel algorithm is faster compared to its baseline. The efficiency is displayed on the right hand side. For definitions of speedup and efficiency, see, Eq. 58.

In Fig. 11, we show how the number of gridpoints grows with the number of continuous state variables in models that are solved with an adaptive sparse grid. As expected, the number of points grows only moderately with the dimension, i.e. $\sim \mathcal{O}(d)$. The running times on the other hand grow faster, because the number of gridpoints is not the only thing that is increasing with the dimension of the problem. The size of the equation systems (cf., Eqs. 51 and 52, or Eqs. 54 and 55, respectively) that have to be solved at each gridpoint also grows linearly in the dimension, implying that the size of the Jacobians that have to be computed grows quadratically. Therefore, the time spent solving the set of nonlinear equations represents the true roadblock of the time iteration scheme presented here. However, we can - at least to some limited extend - control the increasing running times by simply using a larger number of CPUs.

3.5 Performance and Accuracy

So far, we have described the models we solve, the algorithm we use to solve them, and the way in which we parallelize this algorithm. In this section, we show how this algorithm performs in solving the IRBC model. Yet first of all, some implementation details and the measures we use to assess accuracy have to be described.

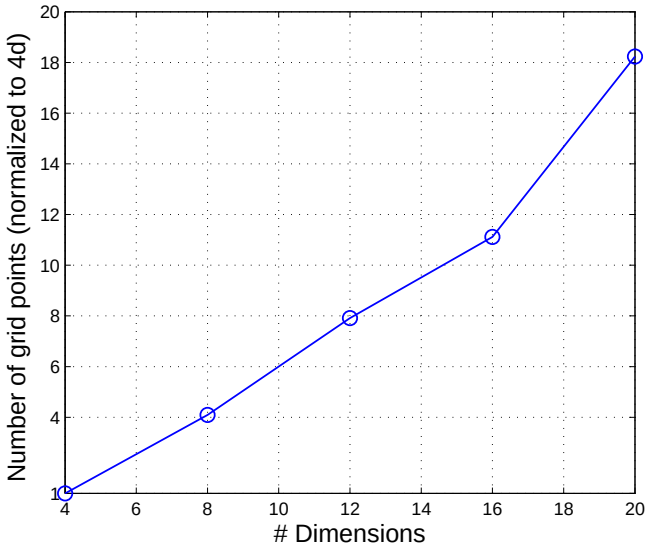

Figure 11: The figure shows how the number of gridpoints (normalized to 4 dimensions) grows with increasing dimensionality of the problem. The test problem is the IRBC model run with an adaptive sparse grid and the refinement criterion set to $|\alpha_{\vec{l},\vec{i}}| \geq \epsilon = 2.5 \cdot 10^{-3}$ (cf., Eq. 32). Note that this choice of ϵ leads to model accuracies comparable to the ones reported in Tab. 4.

Implementation Details

We choose the size of the state space to be big enough such that long simulation paths stay inside the state space for all considered grid structures and dimensions. It turns out that the state space needs to be bigger in the capital dimensions for the non-smooth model than for the smooth one.

Another important detail of the implementation is the integration procedure used to evaluate the expectations operator, e.g. in Eq. 51. As we want to focus on the grid structure, we chose an integration rule that is simple and fast, yet not very accurate.[7] In particular, we use a simple monomial rule that uses just two evaluation points per shock, i.e. $2(N+1)$ points in total (see, [18], with references therein). As we apply the same rule along the time-iteration algorithm as well as for the error evaluation, this choice factors out the question of finding integration procedures that are both accurate and efficient. In principle integration could also be carried out using an (adaptive) sparse grid (see Appendix B), yet not over the same space that the policy functions are interpolated on. Therefore, we view integration as a problem that is orthogonal to the choice of the grid structure, and thus do not focus on it.

Finally, we would like to discuss the possibility of accelerating the time iteration procedure by starting with coarse grids and later changing to finer grids. For instance, using 'classical' sparse grids, the overall computation time can be

[7] However, the integration rule we use delivers the same accuracy as Monte Carlo integration with 1000 random evaluation points.

reduced by one order of magnitude when using a level 1 grid for 200 iterations, followed by 80 iterations on a level 2 grid, before finally using a level 3 grid for 20 periods instead of running all 300 iterations with a grid of level 3. Our tests indicate that this approach yields the same accuracy of results as if the entire simulation would have been carried out at level 3. As with integration, it is not the focus of this paper to discuss the most efficient acceleration strategy. Of course, when we compare results, they are achieved with comparable acceleration strategies.

Error Measure

To evaluate the accuracy of our solutions we compute (unit-free) errors in the $N+1$ equilibrium conditions. Namely, for all countries $j \in \{1, ..., N\}$ we get one Euler equation error:

$$\left[\beta \cdot \mathbb{E}_t \left\{\lambda_{t+1} \cdot \left[a_{t+1}^j \cdot A \cdot \alpha \cdot (k_{t+1}^j)^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2\right)\right]\right\}\right] \cdot \left[\lambda_t \cdot \left(1 + \phi \cdot g_{t+1}^j\right)\right]^{-1} - 1, \quad (59)$$

and we get one additional error from the aggregate resource constraint:

$$\sum_{j=1}^{N} \left(a_t^j \cdot A \cdot (k_t^j)^\alpha + k_t^j \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g_{t+1}^j)^2\right) - k_{t+1}^j - \left(\frac{\lambda_t}{\tau_j}\right)^{-\gamma^j}\right) \cdot \left(\sum_{j=1}^{N} \left(a_t^j \cdot A \cdot (k_t^j)^\alpha + k_t^j \cdot \left(-\frac{\phi}{2} \cdot (g_{t+1}^j)^2\right)\right)\right)^{-1}. \quad (60)$$

These expressions are evaluated by using the computed equilibrium policy function to calculate both today's policy and next period's policy. We compute these errors for all points in the state space that are visited along a (ten thousand period) long simulation path (smooth model) or for ten thousand points drawn from a uniform distribution over the state space (non-smooth model). For each of these points we get $N+1$ errors. We then take the maximum over the absolute value of these errors, which results in one error for each point. Over all these errors, we compute both the maximum (Max. Error) and the average (Avg. Error), which we all report in $\log_{10}$-scale.
In case of the IRBC model with irreversible investment there is one additional complication. Denoting the error defined in Eq. 59 by EE^j and defining the percentage violation of the irreversibility constraint by

$$IC^j \equiv 1 - \frac{k_{t+1}^j}{k_t^j \cdot (1-\delta)} \quad (61)$$

the error is now given by

$$\max\left(EE^j, IC^j, \min\left(-EE^j, -IC^j\right)\right). \quad (62)$$

The first term within the max operator, EE^j, is positive when the marginal cost of investing in country j today is lower then the discounted marginal benefit

Dimension	Level	Points	Max. Error	Avg. Error
4	3	137	-2.82	-3.68
4	4	401	-2.98	-4.19
4	5	1'105	-3.19	-4.24
4	6	2'929	-3.28	-4.55

Table 3: Maximum and average errors of 'classical' sparse grid solutions of the smooth IRBC model with fixed dimension and increasing approximation level. All errors are given in $\log_{10}$-scale. In addition, the number of gridpoints is reported.

Dimension	Level	Points	Max. Error	Avg. Error
4	2	41	-2.80	-3.68
8	2	145	-2.96	-3.25
12	2	313	-2.64	-3.27
16	2	545	-2.59	-3.29
20	2	841	-2.58	-3.29
22	2	1013	-2.60	-3.29
24	2	1201	-2.55	-3.29
4	3	137	-2.82	-3.68
8	3	849	-3.04	-3.83
12	3	2649	-2.71	-3.78
16	3	6049	-2.72	-3.80

Table 4: Maximum and average errors for 'classical' sparse grid solutions of the smooth IRBC model with increasing dimension: up to dimension 24 for level 2, and up to dimension 16 for level 3. All errors are given in $\log_{10}$-scale. Moreover, the number of gridpoints is reported.

of this investment tomorrow. Thus, investment in country j is sub-optimally low. Independent of irreversibility, this is always an error, as the irreversibility constraint does not prohibit investing more. The second term, IC^j, is positive if the irreversibility constraint is violated; in this case, it measures the relative size of the violation. Finally, if $-EE^j$ is positive, then the marginal cost of investing in country j today is higher then the discounted marginal benefit of this investment tomorrow. Thus, investment in country j is sub-optimally high. Thus, lower investment would be optimal. Yet, if the constraint is almost binding investment can only be lowered slightly; in this case, the error is given by the slack in the irreversibility constraint, which is $-IC^j$. Therefore, in the case that $-EE^j$ is positive, the error is not simply given by $-EE^j$ but by $\min\left(-EE^j, -IC^j\right)$.

Results for the smooth IRBC model

We first consider 'classical' sparse grid solutions of the smooth IRBC model. Tab. 3 shows how the approximation errors decrease as we increase the resolution level of the grid keeping the dimensionality of the problem fixed at $2N = 4$. Recall that we report all errors in $\log_{10}$-scale. We can see that the maximal and average errors fall as the level of the grid and thereby the number of gridpoints

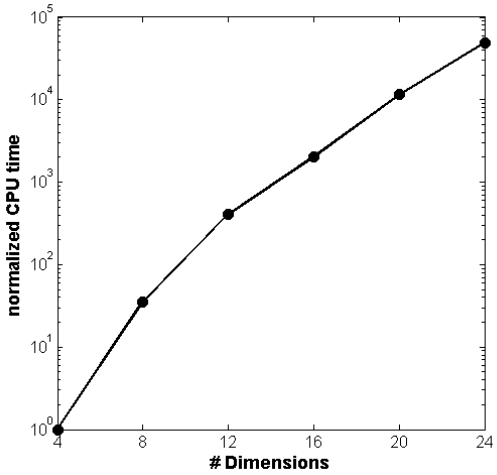

Figure 12: The figure shows how CPU time (in log-scale) increases with the dimensionality of the problem. CPU time is normalized to the computation time of the four dimensional case. The test problem was one representative time-iteration step in a 'classical' sparse grid of level 2.

increases. This is exactly what one would expect. Note that the errors are reasonably low even for a relatively small number of gridpoints. This is simply because the policy functions of the IRBC model are very smooth. Therefore, an adaptive grid cannot improve much on 'classical' sparse grids. For the same reason, the accuracy of our solutions is lower than the accuracy obtained by some smooth approximation methods used in the comparison study summarized by [21].

Let us now consider higher dimensions. In Tab. 4, we vary the dimensionality of the problem while keeping the grid level fixed. We find that the accuracy fluctuates little as we change the dimensionality of the problem.[8] The solutions for four dimensions seem to be somewhat more accurate than for higher dimensions. Importantly, however, there seems to be no clear downward trend in accuracy. Thus, for a given resolution level the accuracy of the solution does not deteriorate as the dimensionality of the problem is increased massively. Clearly, the number of gridpoints increases, but far from exponentially, which is the defining feature of sparse grids. Of course, what matters in the end is not the number of gridpoints needed, but the computation time and memory required. Yet, both time and memory consumption of the algorithm highly depend on the number of gridpoints. Concerning CPU time, Fig. 12 shows that it substantially increases as we consider higher dimensions. This effect, however, is known and was previously reported e.g. by [17, 25], who both were using Smolyak sparse grids with global polynomials. While for example an average time step for $2N = 4$ consumes ~ 0.0001 CPUh using a 'classical' sparse grid of level 2, the same model with $2N = 24$ needs ~ 20 CPUh. This is largely

[8]Some fluctuations are to be expected, especially because the preference heterogeneity depends on the dimension of the problem (see Tab. 2)

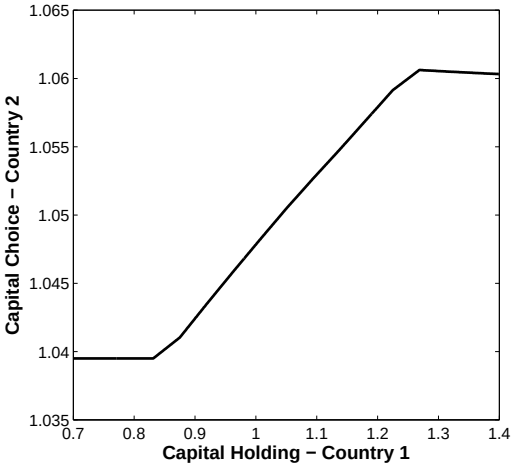

Figure 13: Capital choice of country 2 as a function of capital holding of country 1, where all other 3 state variables of the two country model are kept fixed at their deterministic steady state levels. The 4-dimensional policy function was interpolated on an adaptive sparse grid (with refinement threshold $\epsilon = 0.0033$).

driven by the increasing size of the non-linear equation systems that have to be solved, and only to a lesser extend due to the increasing number of gridpoints. All in all, the increase in CPU time is less than exponential (cf., Fig. 12). Thus, we can - at least to some limited extend - control the increasing running times by simply using a larger number of cores (cf., Sec. 3.4). This stands in strong contrast e.g. to [17, 25], who only provide serial algorithms.

Results for the non-smooth IRBC model

We now turn to the IRBC model with irreversible investment, as described in Sec. 3.2. The assumption that investment is not reversible induces non-differentiabilities in the policy functions that we interpolate. Two such kinks can be observed in Fig. 13. For the two country case, Fig. 13 plots the capital choice (i.e. the end of period capital) for country 2 as a function of the beginning of period capital holding of country 1 while keeping all other state variables fix. If the capital of country 1 is low, then investment opportunities in that country are better than in country 2, thus the irreversibility constraint for country 2 is binding. This explains the flat part of the policy function at the lower left side of Fig. 13. On the other hand, if capital of country 1 is very high, then its irreversibility constraint is binding and thus limits the transfer of resources to country 2. Therefore the policy function is non-increasing at the very right. It is in fact slightly decreasing for consumption smoothing reasons. When looking at Fig. 13, one has to keep in mind that a kink that appears as a single point in that slice through the $2N$-dimensional state space is in fact a $(2N-1)$-dimensional hypersurface. Clearly, such high-dimensional kinks pose a substantial challenge for constructing an accurate global approximation.
In order to gain an understanding of how accurately the (adaptive) sparse grid

Dimension	Level	Points	Max. Error	Avg. Error
4	3	137	-1.50	-2.70
4	4	401	-2.06	-2.86
4	5	1'105	-2.07	-2.96
4	6	2'929	-2.17	-3.07

Table 5: Maximum and average errors of 'classical' sparse grid solutions of the **non-smooth** IRBC model with fixed dimension and increasing approximation level. All errors are given in $\log_{10}$-scale. Furthermore, the number of gridpoints is reported.

ϵ	Points	Max. Error	Avg. Error	Max. Level Reached
0.0150	429	-2.10	-3.05	5 (1'105)
0.0100	510	-2.11	-3.11	6 (2'929)
0.0067	724	-2.43	-3.20	7 (7'537)
0.0050	823	-2.46	-3.19	8 (18'945)
0.0033	1'454	-2.62	-3.27	10 (113'409)

Table 6: Maximum and average errors for an adaptive sparse grid solution of the **non-smooth** IRBC model of different refinement thresholds ϵ are reported. In addition, the size of the respective $2N = 4$-dimensional adaptive sparse grids is reported in the second column. Finally, the last column shows two numbers. The first one is the highest resolution level that was reached for a particular ϵ. The second number (in brackets) indicates how many gridpoints a corresponding 'classical', fixed sparse grid of that resolution level would comprise. All errors are reported in $\log_{10}$-scale.

method is able to solve high-dimensional non-smooth IRBC models, we first focus on the 4-dimensional case.[9] Tab. 5 reports errors for 'classical' sparse grids of increasing resolution levels. It reveals two important insights. First, both the maximum and the average errors of non-smooth IRBC models are considerably worse than for smooth IRBC models of comparable resolution (cf., Tab. 3). The reason is that a relatively coarse grid can hardly capture kinks. Thus, interpolation performs particularly poorly close to non-differentiabilities. The second insight is that increasing the global resolution level initially decreases the errors substantially, but from level four onward, the improvements are only very modest. In particular, the maximum error barely decreases as the number of gridpoints increases. The reason for this behavior is that even with a relatively high global resolution, the local resolution is not sufficiently high to at least roughly match the kinks.

Having these findings in mind, we now turn our attention from the 'classical' sparse grids to adaptive sparse grids. Tab. 6 and Fig. 14 show that adaptive sparse grids are more efficient in reducing the approximation errors, as they put additional resolution where needed, while not wasting resources in areas of smooth variation. For instance, an adaptive sparse grid with 429 points reaches errors comparable to a fixed sparse grid with 2'929 points – or an adaptive grid with 724 points already outperforms the previously mentioned fixed grid, even

[9]For the 4-dimensional non-smooth model, we not only use ten thousand testpoints, but one hundred thousand (cf., Tabs. 5, 6, and Fig. 14).

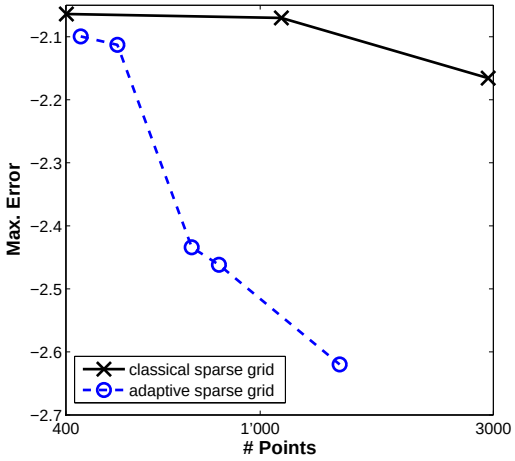

Figure 14: Comparison of the maximum error (in $\log_{10}$-scale) for conventional and adapive sparse grid solutions to the $2N = 4$-dimensional **non-smooth** IRBC model as a function of gridpoints. The datapoints for the 'classical' sparse grids stem from models run with fixed levels 4 to 6. The respective adaptive sparse grid solutions arise from varying refinement thresholds ϵ ranging from 0.01 down to 0.0033. For similar sized grids, the 'classical' sparse grid (solid line with crosses indicating data points) is much less accurate than the adaptive sparse grid (dashed line with circles indicating data points).

though the latter consists of four times more gridpoints (see, Tab. 5).
In Tab. 6, we report results for different refinement thresholds ϵ. The smaller the chosen refinement threshold ϵ, the larger the maximum refinement level reached and the larger the number of gridpoints. When the threshold is lowered from 0.015 to 0.0033, the number of gridpoints approximately triples, improving the maximum error by a factor of $10^{-2.10}/10^{-2.62} \approx 3.3$. This is in stark contrast to the results reported in Tab. 5, where we find that increasing the size of 'classical' sparse grids improves the maximum error only very slowly. With the lowest threshold considered (0.0033), the adaptive sparse grid algorithm continues to refine until level 10. Thus, we are able to locally mimic an interpolant that is of the same order of approximation as a 'classical' sparse grid of level 10. While the adaptive sparse grid consists of $1'454$ gridpoints, the latter grid would comprise more than one hundred thousand points (cf., the last column of Tab. 6). This comparison shows that the adaptive sparse grid introduces an additional layer of sparsity on top of the a priori sparse structure of the 'classical' sparse grid. Making use of this sparsity, additional gridpoints can reduce the approximation error much more efficiently, as one can see in Fig. 14. This is possible as gridpoints are placed only where high resolution is needed while just few points are put in areas where the policies to be approximated vary little. In order to illustrate this feature, we display in Fig. 15 slices of 4-dimensional grids, one 'classical' of fixed refinement level 6 (cf., Tab. 3) and the other 'adaptive' with threshold $\epsilon = 0.0033$ (cf., Tab. 6). In spite of having less than half as many points, the solution provided by the adaptive sparse grid is much more accurate.

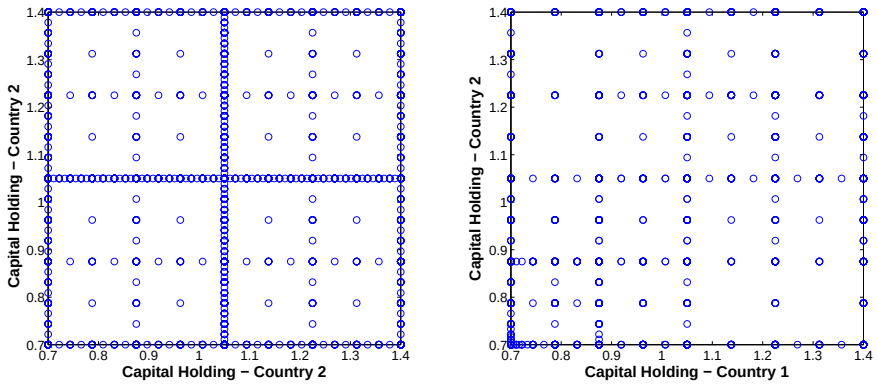

Figure 15: This figure displays 2-dimensional projections of two different grids. The left one is from a 'classical' sparse grid of level 6 (2'929 points), the right one from an adaptive grid with refinement threshold $\epsilon = 0.0033$ (1'454 points). Both grids were generated in the course of running a $2N = 4$ dimensional simulation. The x-axis shows capital holding of country 1, the y-axis shows capital holding of country 2, while the productivities of the two countries are kept fixed at their unconditional means.

From the figure, one can see that this is achieved mainly by adding gridpoints in the lower left corner of the graph. However, note that the actual grid is 4-dimensional, thus the 2-dimensional projection can only give a rough idea of the sparse grid structure.

Let us now turn to higher-dimensional models. Tab. 7 reports errors for adaptive sparse grids of a fixed refinement threshold $\epsilon = 0.01$ and increasing dimensionality.[10] We find that the accuracy weakly depends on the dimension. There seems to be a moderate downward trend both in the maximum as well as in the average error. This behavior is not surprising as the kinks, being $2N - 1$ dimensional objects, become much harder to approximate as N increases. Also, the maximum refinement level $L_{max} = 6$ is binding for dimensions six and eight, while it is not binding for dimension four. Therefore, with a higher L_{max} the errors for higher dimensions would slightly improve relative to dimension four. Another way to counteract the worsening errors would be to lower ϵ moderately when the dimension of the problem is increased. However, note that even for the 8-dimensional model only very few errors are close to the maximum error, as shown in Fig. 16. There, one can see that the 99% quantile of the error distribution is already better than minus two. Regarding solution time, it is clear that the non-smooth IRBC models pose a considerably larger burden compared to the smooth IRBC models. Thus, for a given dimensionality, the CPU time spent solving for an equilibrium is considerably larger for the non-smooth model. One reason is that the nonlinear system of equations to be solved is of size $2N + 1$ (cf., Eqs. 54 and 55), compared to $N + 1$ in case of the smooth IRBC models (cf., Eqs. 51 and 52). On top of this, the (local) resolution needs

[10]Note that we have chosen $\epsilon = 0.01$ due to the fact that this setting seems to provide decent errors at moderate costs (cf., Tab. 3).

Dimension	Points	Max. Error	Avg. Error	$\lvert V_6^{S,CC}\rvert$
4	510	-2.15	-3.11	2'929
6	1'950	-1.87	-2.71	15'121
8	5'643	-1.63	-2.51	56'737

Table 7: Errors for an adaptive sparse grid solution of the **non-smooth** IRBC model with increasing dimension. For all dimensions, we start with a 'classical' sparse grid of level three and use a refinement threshold $\epsilon = 0.01$ to further refine the grid up to a maximum level of six. The number of required gridpoints of the adaptive sparse grid solution is contrasted by the corresponding grid size of the 'classical' sparse grid. All errors are reported in $\log_{10}$-scale.

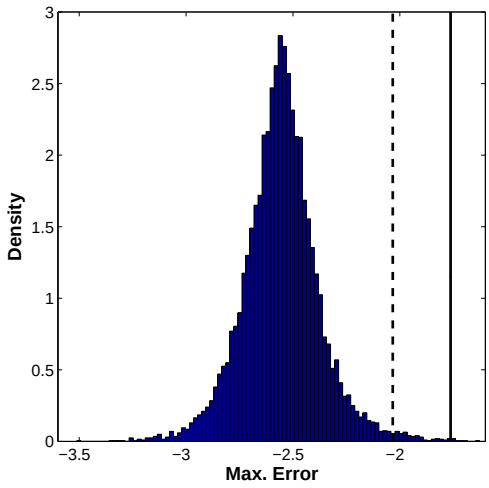

Figure 16: Distribution of maximum approximation errors (in $\log_{10}$-scale) for the non-smooth IRBC model with eight-dimensional state space (four countries). The 99% quantile is indicated by the dashed line, while the solid line corresponds to the $99,9\%$ quantile.

to be considerably higher in order to achieve decent errors (cf., Tab. 6), resulting in more gridpoints when comparing the smooth and non-smooth IRBC models of fixed dimension. Both factors increase the total degrees of freedeom of the problem substantially ($13 \cdot 1'201 = 15'613$ for the 24-dimensional smooth model versus $9 \cdot 5'643 = 50'787$ for the 8-dimensional non-smooth model). The required CPU time for a representative time iteration step in a two country model now consumes ~ 0.003 CPUh, while a 8-d model may require up to ~ 250 CPUh. However, this is still not a roadblock for solving such complex models. All in all, CPU time again increases less than exponentially in the dimension of the problem. Therefore, we can still compute solutions of high-dimensional non-smooth models in reasonable time by using acceleration methods (see, Sec. 3.5) and employing a large number of cores (see, Sec. 3.4). Hence, we can obtain results within one day, even for the largest non-smooth models considered here.

To sum up, the presented adaptive sparse grid method can successfully com-

pute global solutions of high-dimensional, non-smooth dynamic models. Resolving such non-smooth behavior is not possible with the methods for high-dimensional models that have so far been established in economics (see, e.g. [19, 22]). On the other hand, methods that are designed to handle non-differentiabilities (see, e.g. in [15, 3, 9, 1]) are only capable of solving models with up to three continuous state variables. Therefore, adaptive sparse grids seem to be the method of choice for problems that are both non-smooth and high-dimensional.

4 Conclusion

We embed an adaptive sparse grid algorithm in a time-iteration procedure to solve dynamic economic models. In addition, we provide a fully hybrid parallel implementation of the resulting time-iteration adaptive sparse grid algorithm. With this implementation, we can efficiently use current high-performance computing technology and are, to the best of our knowledge, the first paper on dynamic economic models to do so.

The time-iteration adaptive sparse grid algorithm we use is highly flexible and scalable. First, by choosing the resolution level we can tightly control accuracy, thus being able to strike the right balance between running times and the desired accuracy of the solution. Second, due to the highly parallelized implementation, we can speed up the computations tremendously by simply using a larger number of CPUs. This allows us to solve hard problems in a relatively short amount of time, and to tackle problems that were so far non-tractable.

We apply this algorithm to an IRBC model with adjustment costs and up to 24 continuous state variables. We are also able to solve a high-dimensional IRBC model with irreversible investment. In that application the comparative advantage of the adaptive sparse grid comes into full play, as it can efficiently capture the kinks induced by irreversibility without wasting additional grid-points in regions of the state space where they are not needed.

Note that to solve the IRBC model we embed the adaptive sparse grid in a time iteration procedure that operates in the space of policy function. However, adaptive sparse grids are also very promising for interpolating value functions in classical dynamic programming applications. For such applications, one only needs to approximate a one-dimensional value function on the adaptive sparse grid, whereas we have to approximate several policy functions on a single grid.

We hope that our paper inspires many economic applications of adaptive sparse grids. Being scalable and flexible, adaptive sparse grids can make use of modern high-performance computing infrastructure, and they can be applied to a broad variety of setups where high dimensional functions have to be interpolated efficiently. This tool thus offers the promise to economic modellers of being able to solve models that include much more heterogeneity than was previously possible.

Appendix

A Sparse Grids with Non-Zero Boundaries

In Sec. 2, we have assumed that the functions under consideration vanish at the boundary of the domain, i.e. $f|_{\partial\Omega} = 0$. To allow for non-zero values at the boundary, the procedure one usually follows is to add additional gridpoints located directly on $\partial\Omega$, and associated basis functions [28, 20]. Doing this naively, one needs at least 3^d gridpoints, which makes the approach inapplicable to high-dimensional problems [20]. In what follows, we discuss two procedures that mitigate this problem.

One way to handle non-zero boundaries even for high-dimensional problems is the so-called 'Clenshaw-Curtis' sparse grid $V_n^{S,CC}$ with equidistant support nodes [24, 20, 2, 27]. The crucial idea is to have only one gridpoint at the lowest level of approximation. Technically, the difference to the sparse grid $V_{n,0}^S$ is just that the index set of the support nodes is not given by Eq. 17, but rather by

$$I_{\vec{l}} := \begin{cases} \{\vec{i} : i_t = 1, 1 \leq t \leq d\} & \text{if } l = 1 \\ \{\vec{i} : 0 \leq i_t \leq 2,\ i_t \text{ even}, 1 \leq t \leq d\} & \text{if } l = 2 \\ \{\vec{i} : 1 \leq i_t \leq 2^{l_t - 1} - 1,\ i_t \text{ odd}, 1 \leq t \leq d\} & \text{else} \end{cases} \tag{63}$$

and the one-dimensional basis functions are given by

$$\phi_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 1 - 2 \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 0 \\ \left\{ \begin{array}{ll} 2 \cdot x_t - 1 & \text{if } x_t \in \left[\frac{1}{2}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 2 \\ \phi_{l,i}(x_t) & \text{else.} \end{cases} \tag{64}$$

The support nodes of the 'Clenshaw-Curtis' grid are shown in Figs. 17 and 18, whereas the basis functions are displayed in Fig. 19. Note that $\phi_{l,i}(x)$ is given by Eq. 7. It is also worth mentioning that the number of gridpoints of the Clenshaw-Curtis grid $|V_n^{S,CC}|$ grows even slower than $|V_{0,n}^S|$ (see, Tab. 8).

Another way to handle $f|_{\partial\Omega} \neq 0$ in high dimensions is to omit gridpoints on the boundary altogether and instead to modify the interior basis functions to extrapolate towards the boundary of the domain. This approach is especially well suited in settings where high accuracy close to the boundary is not required and where the underlying function does not change too much towards the boundary [28].

An appropriate choice for the modified one-dimensional basis functions is

$$\phi_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2^l \cdot x_t + 1 - i & \text{if } x_t \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1 \\ \phi\left(x_t \cdot 2^l - i\right) & \text{else,} \end{cases} \tag{65}$$

where the support nodes have the same coordinates as in the ordinary sparse grid (cf., Eq. 28). The d-dimensional basis functions are obtained in analogy

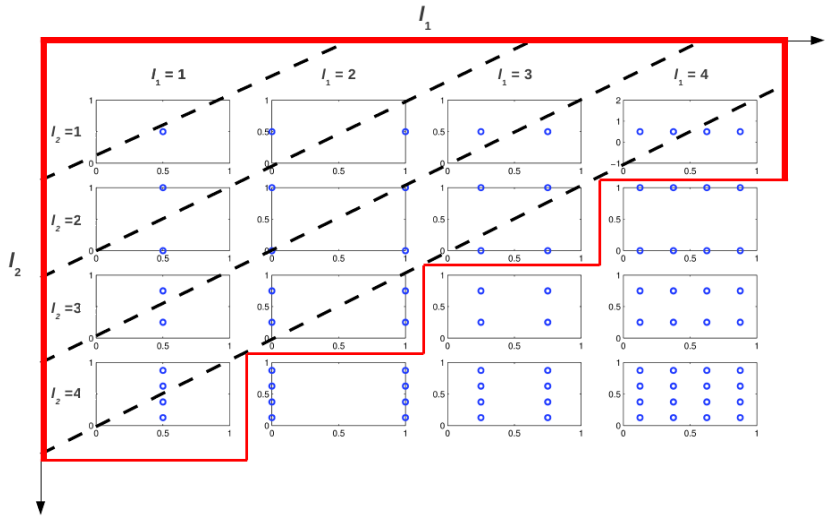

Figure 17: Schematic construction of a level 4 'Clenshaw-Curtis' sparse grid $V_4^{S,CC}$ in two dimensions (cf., Eq. 63). $V_4^{S,CC}$ consists of the hierarchical increment spaces $W_{(l_1,l_2)}$ for $1 \leq l_1, l_2 \leq n = 4$. The area enclosed by the red bold lines marks the region where $|\vec{l}| \leq n + d - 1$, fulfilling Eq. 28. The blue dots represent the gridpoints of the respective subspaces. Finally, the dashed black lines indicate the hierarchical increment spaces for constant $|\vec{l}|$.

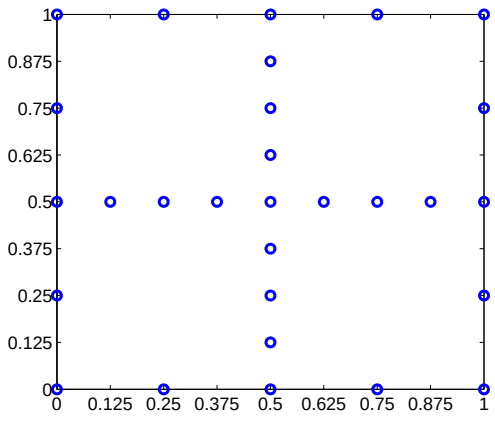

Figure 18: Sparse grid space $V_4^{S,CC}$ in 2 dimensions, constructed according to Eq. 28 from the increments displayed in Fig. 17. Note that it contains only 29 support nodes, whereas a full grid would consist of 225 points.

to Eq. 15 by a tensor product construction of the one-dimensional ones. We denote this function space by V_n^S, as it allows for non-zero boundaries.

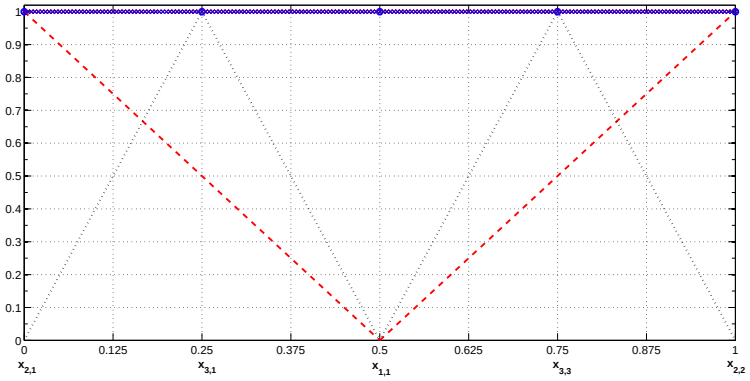

Figure 19: Hierarchical basis functions of the 'Clenshaw-Curtis' grid in one dimension. Level 1 (solid blue), level 2 (dashed red), and level 3 (dotted black).

d	$\lvert V_4 \rvert$	$\lvert V_{0,4}^S \rvert$	$\lvert V_4^{S,CC} \rvert$
1	15	15	9
2	225	49	29
3	3'375	111	69
4	50'625	209	137
5	759'375	351	241
10	$5.77 \cdot 10^{11}$	2'001	1'581
15	$4.37 \cdot 10^{17}$	5'951	5'021
20	$3.33 \cdot 10^{23}$	13'201	11'561
30	$1.92 \cdot 10^{35}$	41'601	37'941
40	$1.11 \cdot 10^{47}$	95'201	88'721
50	$6.38 \cdot 10^{58}$	182'001	171'901
100	>Googol	1'394'001	1'353'801

Table 8: Number of gridpoints for several different grid types of level 4. First column: dimension; second column: full grid; third column: 'classical' L_2 optimal sparse grid with no points at the boundaries; last column: 'Clenshaw-Curtis' sparse grid.

B Hierarchical Integration

The sparse grid approach can also be used for high-dimensional numerical integration, e.g. for the computation of expectations [12, 24, 27].

Starting from Eq. 30, the expected value of the interpolant can be evaluated as follows:

$$\mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \int_\Omega \phi_{\vec{l},\vec{i}}(\vec{x}) d\vec{x}, \tag{66}$$

where we assume for simplicity that the probability density is 1 on $\Omega = [0,1]^d$. Using the basis functions described in Eqs. 63 and 64 as an example, the one

dimensional integral can now easily be computed analytically [24] starting from

$$\int_0^1 \phi_{l,i}(x)\,dx = \begin{cases} 1, & \text{if } l = 1 \\ \frac{1}{4} & \text{if } l = 2 \\ \\ 2^{1-l} & \text{else.} \end{cases} \tag{67}$$

The multi-dimensional integrals are simply given by the product of one dimensional integrals. Following [24], we denote $\int_\Omega \phi_{l,i}(\vec{x})\,d\vec{x} = J_{\vec{l},\vec{i}}$, and can thus rewrite Eq. 66 by

$$\mathbb{E}[u(\vec{x})] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot J_{\vec{l},\vec{i}}. \tag{68}$$

Eq. 68 states that the expectation is given by the arithmetic sum over all grid-points of the product between the hierarchical surpluses and the integral weights.

References

[1] Francisco Barillas and Jesús Fernández-Villaverde. A generalization of the endogenous grid method. *Journal of Economic Dynamics and Control*, 31(8):2698 – 2712, 2007.

[2] Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12:273–288, 2000.

[3] J. Brumm and M. Grill. Computing equilibria in dynamic models with occasionally binding constraints. *Journal of Economic Dynamics and Control, accepted for publication.*

[4] H.-J. Bungartz and S. Dirnstorfer. Multivariate quadrature on adaptive sparse grids. *Computing*, 71:89–114, 2003.

[5] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.

[6] Yongyang Cai, Kenneth L. Judd, Greg Thain, and Stephen J. Wright. Solving dynamic programming problems on a computational grid. NBER Working Papers 18714, National Bureau of Economic Research, Inc, January 2013.

[7] Wouter J. Den Haan, Kenneth L. Judd, and Michel Juillard. Computational suite of models with heterogeneous agents ii: Multi-country real business cycle models. *Journal of Economic Dynamics and Control*, 35(2):175–177, February 2011.

[8] J. J. Dongarra and A. J. van der Steen. High-performance computing systems: Status and outlook. *Acta Numerica*, 21:379–474, 4 2012.

[9] Giulio Fella. A generalized endogenous grid method for non-smooth and non-concave problems. *Review of Economic Dynamics, In Press*, 2013.

[10] J. Garcke and M. Griebel. *Sparse Grids and Applications.* Lecture Notes in Computational Science and Engineering Series. Springer-Verlag GmbH, 2012.

[11] Alan Genz. Testing multidimensional integration routines. In *Proc. of international conference on Tools, methods and languages for scientific and engineering computation*, pages 81–94, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[12] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18:209–232, 1998.

[13] M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998. also as Proceedings Large-Scale Scientific Computations of Engineering and Environmental Problems, 7. June - 11. June, 1997, Varna, Bulgaria, Notes on Numerical Fluid Mechanics 62, Vieweg-Verlag, Braunschweig, M. Griebel, O. Iliev, S. Margenov and P. Vassilevski (editors).

[14] C. Hager, S. Hüeber, and B. Wohlmuth. Numerical techniques for the valuation of basket options and its greeks. *J. Comput. Fin.*, 13(4):1–31, 2010.

[15] Thomas Hintermaier and Winfried Koeniger. The method of endogenous gridpoints with occasionally binding constraints among endogenous variables. *Journal of Economic Dynamics and Control*, 34(10):2074–2088, October 2010.

[16] Gabriele Jost, Barbara Chapman, and Ruud van der Pas. *Using OpenMP - Portable Shared Memory Parallel Programming.* MIT Press, 2007.

[17] Kenneth Judd, Lilia Maliar, Rafael Valero, and Serguei Maliar. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. Working Papers. Serie AD 2013-06, Instituto Valenciano de Investigaciones Económicas, S.A. (Ivie), September 2013.

[18] Kenneth L Judd. *Numerical methods in economics.* The MIT press, 1998.

[19] Michel Juillard and Sébastien Villemot. Multi-country real business cycle models: Accuracy tests and test bench. *Journal of Economic Dynamics and Control*, 35(2):178–185, February 2011.

[20] Andreas Klimke and Barbara Wohlmuth. Algorithm 847: Spinterp: piecewise multilinear hierarchical sparse grid interpolation in matlab. *ACM Trans. Math. Softw.*, 31(4):561–579, December 2005.

[21] Robert Kollmann, Serguei Maliar, Benjamin A. Malin, and Paul Pichler. Comparison of solutions to the multi-country real business cycle model. *Journal of Economic Dynamics and Control*, 35(2):186 – 202, 2011. Computational Suite of Models with Heterogeneous Agents II: Multi-Country Real Business Cycle Models.

[22] Dirk Krueger and Felix Kubler. Computing equilibrium in OLG models with stochastic production. *Journal of Economic Dynamics and Control*, 28(7):1411 – 1436, 2004.

[23] Deborah J. Lucas. Asset pricing with undiversifiable income risk and short sales constraints: Deepening the equity premium puzzle. *Journal of Monetary Economics*, 34(3):325 – 341, 1994.

[24] Xiang Ma and Nicholas Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *J. Comput. Phys.*, 228(8):3084–3113, May 2009.

[25] Benjamin A Malin, Dirk Krüger, and Felix Kübler. Solving the multi-country real business cycle model using a smolyak-collocation method. *Journal of Economic Dynamics and Control*, 35(2):229–239, October 2010.

[26] Alin Murarasu, Josef Weidendorfer, Gerrit Buse, Daniel Butnaru, and Dirk Pflüeger. Compact data structure and parallel alogrithms for the sparse grid technique. In *16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2011.

[27] Erich Novak and Klaus Ritter. High dimensional integration of smooth functions over cubes. *Numerische Mathematik*, 75:79–97, 1996.

[28] Dirk Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. PhD thesis, München, August 2010.

[29] Dirk Pflüger, Benjamin Peherstorfer, and Hans-Joachim Bungartz. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity*, 26(5):508—-522, October 2010. published online April 2010.

[30] Rolf Rabenseifner, Georg Hager, and Gabriele Jost. Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In *Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, PDP '09, pages 427–436, Washington, DC, USA, 2009. IEEE Computer Society.

[31] Thomas Rauber and Gudula Rünger. *Parallel Programming: for Multicore and Cluster Systems*. Springer, 2010 edition, March 2010.

[32] Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Rev.*, 50(1):91–112, February 2008.

[33] Olaf Schenk, Andreas Wächter, and Michael Hagemann. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Comput. Optim. Appl.*, 36(2-3):321–341, April 2007.

[34] Anthony Skjellum, William Gropp, and Ewing Lusk. *Using MPI*. MIT Press, 1999.

[35] S.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963.

[36] Chris I. Telmer. Asset-pricing puzzles and incomplete markets. *The Journal of Finance*, 48(5):1803–1832, 1993.

[37] Andreas Waechter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, May 2006.

[38] Viktor Winschel and Markus Kraetzig. Solving, estimating, and selecting nonlinear dynamic models without the curse of dimensionality. *Econometrica*, 78(2):803–821, 2010.

[39] Christoph Zenger. Sparse grids. In Wolfgang Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251. Vieweg, 1991.